IMPLEMENTASI SISTEM PENDETEKSI ROGUE ACCESS POINT DENGAN METODE PERHITUNGAN NILAI ROUND TRIP TIME

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Fahmi Syahrulah
NIM : 115060800111081

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2018

PENGESAHAN

IMPLEMENTASI SISTEM PENDETEKSI ROGUE ACCESS PPOINT DENGAN METODE PERHITUNGAN NILAI ROUND TRIP TIME

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Fahmi Syahrulah
NIM : 115060800111081

Skripsi ini telah diuji dan dinyatakan lulus pada
03 Agustus 2018
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Adhitya Bhawiyuga, S.Kom, M.Sc	Mahendra Data, S.Kom, M.Kom
NIP : 198907202018031002	NIK : 2015038611171001

Mengetahui
Ketua Jurusan **Teknik Informatika**

Tri Astoto Kurniawan, S.T, M.T, Ph.D
NIP: 19710518 200312 1 001

Identitas Tim Penguji

Penguji 1 :

Nama : Ir. Primanantara Hari Trisnawan, M. Sc.

NIP/NIK : 19680912 199403 1 002

Penguji 2 :

Nama : Fariz Andri Bakhtiar, S.T., M.Kom.

NIP/NIK : 85031406110028

PERNYATAAN ORISINALITAS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini dan disebutkan dalam daftar pustaka.

Apabila ternyata didalam naskah skripsi ini dapat dibuktikan terdapat unsur-unsur *plagiasi, saya bersedia skripsi ini digugurkan* dan gelar akademik yang telah saya peroleh (sarjana) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 7 Agustus 2018

Fahmi Syahrulah

NIM : 115060800111081

Daftar Riwayat Hidup

Nama : Fahmi Syahrulah

Tempat, Tanggal lahir : Mojokerto, 3 September 1992

Riwayat Pendidikan : SDN Wates 5, Mojokerto

SMPN 1 Kota Mojokerto

SMAN 3 Kota Mojokerto

KATA PENGANTAR

Puji syukur kehadirat Allah SWT karena atas rahmat dan hidayah-Nya, penulis dapat menyelesaikan skripsi dengan judul "**Implementasi Sistem Pendeteksi Rogue Access Point dengan Metode Perhitungan Nilai Round Trip Time**".

Penyusunan skripsi ini tidak lepas dari bantuan semua pihak yang telah memberikan semangat, doa, bimbingan, kritik, serta saran. Maka dari itu penulis menyampaikan ucapan terimakasih kepada:

1. Seluruh keluarga saya. Bapak, Ibu dan kedua kakak saya yang telah membantu dan mendukung demi kelancaran pengerjaan skripsi ini.
2. Bapak Wayan Firdaus Mahmudy, S.Si, M.T, Ph.D selaku Dekan Fakultas Ilmu Komputer Universitas Brawijaya, Tri Astoto Kurniawan, S.T, M.T, Ph.D selaku Ketua Jurusan Teknik Informatika Universitas Brawijaya dan Bapak Agus Wahyu Widodo, S.T, M.Sc. selaku Ketua Prodi Teknik Informatika
3. Bapak Adhitya Bhawiyuga, S.Kom, M.Sc. selaku dosen pembimbing I yang telah membimbing dan mengarahkan penulis sehingga dapat menyelesaikan skripsi ini.
4. Bapak Mahendra Data, S.Kom, M.Kom. selaku dosen pembimbing II yang telah banyak memberikan ilmu dan saran untuk skripsi ini.
5. Semua rekan-rekan mahasiswa teknik informatika 2011. Teman-teman kontrakan Bu Sumi, teman kos Betek dan Bantaran dan Keluarga Kampung Melati.
6. Semua pihak yang tidak dapat penulis sebutkan satu per satu yang terlibat baik secara langsung maupun tidak langsung demi terselesaikannya tugas akhir ini.

Semoga skripsi ini dapat bermanfaat dan berguna bagi pembaca terutama mahasiswa Prodi Teknik Informatika Jurusan Teknik Informatika Universitas Brawijaya.

Malang, 7 Agustus 2018

Penulis

Fsyahrulah@gmail.com

ABSTRAK

Teknologi jaringan nirkabel atau *wireless* yang semakin berkembang membuat semakin banyak pula kemungkinan akan serangan pada jaringan nirkabel. Hal ini membahayakan bagi keamanan data *user*. Salah satu serangan yang memanfaatkan jaringan *wireless* yakni *Rogue access point*. *Rogue access point* merupakan sebuah serangan dengan membuat sebuah *access point* palsu dengan melakukan imitasi SSID dari *access point* yang sebenarnya atau *legitimate access point*. Untuk mendeteksi RAP pada sisi *client* dibuat sistem pendeteksian RAP dengan metode perhitungan nilai *round trip time*. Sistem pendeteksian ini melakukan pengiriman paket DNS *lookup* untuk mendapatkan nilai *round trip time*. RAP akan menghasilkan nilai *round trip time* yang lebih besar daripada *legitimate access point* akibat dari pertambahan jumlah hop yang dilaluinya. Untuk meminimalisir dampak dari dari kondisi jaringan yang tidak stabil yang dapat mempengaruhi akurasi pendeteksian, dibuat mekanisme *filtering* untuk menyaring data *round trip time* yang anomali. Pengujian dilakukan dengan mengirim DNS *lookup* pada 50 *domain* yang berbeda. Pengujian dilakukan pada kedua *access point* identik yang salah satunya merupakan *rogue access point*. Perbandingan nilai *round trip* time yang didapatkan dari kedua *access point* akan digunakan sebagai parameter dalam menentukan *rogue access point*. Proses *filtering* mampu menghapus nilai anomali saat dilakukan pengujian simulasi jaringan lambat dengan menambah *delay* dengan menggunakan *NETEM*. Pengujian pertama yang telah dilakukan pada suatu jaringan A dari 100 kali pengujian didapatkan hasil akurasi 90%. Terdapat 10 kali program salah dalam melakukan pendeteksian. Pengujian kedua pada jaringan B dari 100 kali pengujian didapatkan akurasi 95% atau hanya ada 5 kali kesalahan pendeteksian. Pengujian ketiga dilakukan dengan mencatat nilai rata-rata *round trip time* sebelum dilakukan *filtering* dan sesudah melakukan *filtering*. Akurasi yang didapatkan meningkat dari 95% menjadi 97% dampak dari proses *filtering* yang dilakukan. Akurasi pendeteksian tidak pernah kurang dari 90%.

Kata kunci : *Rogue access point*, *round trip time*, jaringan nirkabel, serangan jaringan nirkabel

ABSTRACT

With the increasing development of wireless technology, which provoke the cyberattack using wireless technology increase aswell, this would make user data be put to risk. One of the cyberattack that exploit wireless technology is Rogue Access Point. Rogue Access Point (RAP) is an attack that would mimic a real access point, it would create a fake access point with the same SSID as the legitimate access point. To prevent this, RAP detection system was made with comparing round trip time method. This system sent a DNS lookup to get round trip time. RAP would return much longer round trip time value than the legitimate access point, this happen because the added hop that was passed during the trip to RAP. To minimalize the impact from unstable network that would affect the accuracy of detection system, filtering mechanism is used to filter round trip time value that is deemed as an anomaly. Testing will be done to the two similar access point which one of them is the RAP. Round trip time value from both access point will be compared, which will be used as parameter to determine RAP. Filtering process able to filter out the anomaly when tested in a slow network simulation with added delay using NETEM. The first test that is done at network A, the accuracy that we got from 100 test is 90%, 10 problems were found form the first test. The second test done at network B, the accuracy that we got from 100 test is 95%, which means only 5 problems were found. The third test is to record the average value from round trip time before and after filtering process was used. The accuracy that we got increase from 95% to 97% after filtering process. Accuracy detection would be never less than 90%.

Key word : Rouge access point, round trip time, wireless, wireless attack

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB 1 PENDAHULUAN

1.1 Latar belakang

Belakangan ini, teknologi jaringan nirkabel berkembang dengan pesat. Teknologi jaringan nirkabel menjadi teknologi yang paling populer dibandingkan jaringan dengan kabel. Hal ini dikarenakan teknologi jaringan nirkabel lebih fleksibel, mencakup jangkauan yang luas (*scalability*), dan lebih mudah dalam melakukan instalasi (Somayeh, Azizah, Mazdak dan Maziar, 2012). Banyak instansi, kafe maupun tempat umum lainnya yang menggunakan teknologi *wireless* untuk menyediakan akses internet atau yang sering disebut Hotspot atau Wi-Fi.

Namun kemudahan akan akses internet melalui jaringan nirkabel ini lebih banyak menimbulkan permasalahan dalam hal keamanan. Celah keamanan jaringan nirkabel ini sering dimanfaatkan oleh peretas. Beberapa resiko umum yang ditimbulkan dari celah keamanan teknologi jaringan nirkabel antara lain: 1) Pengguna yang tidak terotorisasi bisa saja mendapat akses ke sistem dan mendapat informasi; 2) Tidak adanya integritas atau bisa saja data yang melewati jaringan nirkabel dirusak atau diubah.

Banyak diantara orang yang tidak bertanggung jawab celah keamanan dalam jaringan nirkabel untuk melakukan serangan. Serangan pada jaringan nirkabel pada umumnya dibagi menjadi dua yaitu serangan pasif dan aktif. Dalam serangan pasif pengguna yang tidak terotorisasi mendapat akses ke suatu informasi secara ilegal namun tidak mengubah informasi tersebut. Contoh serangan pasif adalah *eavesdropping* dan *traffic analysis*. Sementara serangan aktif yaitu *user* yang tidak terotorisasi mendapat akses suatu informasi secara ilegal dan memodifikasi informasi tersebut. Contoh serangan aktif adalah *masquerading*, *message modification*, *replay* dan *denial-of-service*(Karygiannis dan Owens, 2002).

Salah satu serangan aktif yang membahayakan adalah *rogue access point*. *Rogue access point* adalah *access point* ilegal yang tidak terotentifikasi dan tidak dipasang oleh administrator WLAN(Han, Sheng, Tan, Li dan Lu, 2011). Pengetahuan dan tingkat kewaspadaan dari pengguna jaringan nirkabel yang berbeda-beda seringkali dimanfaatkan peretas untuk mengelabuhi pengguna jaringan nirkabel agar menyambungkan perangkatnya pada *rogue access point*. *Rogue access point* seringkali dikonfigurasi agar menyerupai *access point* yang sah. Banyak pengguna jaringan nirkabel yang lebih memilih koneksi Wi-Fi dengan sinyal yang kuat tanpa memperhitungkan keabsahan *access point* tersebut(Somayeh, Azizah, Mazdak dan Maziar, 2012). Ketika pengguna jaringan nirkabel terhubung ke *rogue access point* lalu lintas data pengguna otomatis akan ditangkap dan bisa jadi akan disalahgunakan untuk kepentingan pribadi peretas. Untuk itu diperlukan suatu sistem yang mampu melakukan deteksi *rogue access point*.

Deteksi *rogue access point* dibagi menjadi 2 jenis. Yakni dari sisi *admin* dan dari sisi *client*. Deteksi dari sisi *admin* seringkali tidak efektif karena harus melibatkan administrator jaringan. Sementara dari sisi *client* diperlukan metode yang cepat dalam menentukan *access point* yang tersedia dalam suatu lingkungan tersebut adalah *access point* yang terlegitimasi atau *rogue access point*. Penelitian sebelumnya dengan judul "*A Novel Approach for Rogue Access Point Detection on the Client-Side*" oleh S. Nisbakh, deteksi *rogue access point* dilakukan dengan melakukan *traceroute* kepada 2 *access point* yang identik. *Access point* dengan jumlah *hop* lebih banyak akan dideteksi sebagai *rogue access point*. Hal ini dikarenakan *rogue access point* yang diimplementasikan dengan memanfaatkan koneksi dari *legitimate access point* dan memberikan koneksi ilegal menggunakan *wifi card* eksternal akan mempunyai lebih banyak jumlah *hop* daripada *legitimate access point*. Namun *traceroute* seringkali mengembalikan nilai *timeout* dan memunculkan tanda *asterisk* (***) sampai jumlah *hop* mencapai maksimal. Selain itu dalam melakukan *tracerouting* dibutuhkan waktu yang cukup lama.

Oleh karena itu penulis akan meneliti tentang "IMPLEMENTASI PENDETEKSIAN ROGUE ACCESS POINT DENGAN MENGGUNAKAN METODE PERHITUNGAN ROUND TRIP TIME". Deteksi *rogue access point* dengan menghitung *round trip time* pada *access point* yang mempunyai kesamaan SSID dan *MAC address*. Dengan menggunakan nilai *round trip time* dari paket pengujian berupa DNS *request-respons*. Nilai *round trip time* digunakan untuk menghitung kinerja masing-masing *access point* yang terlegitimasi dan *rogue access point*. Hasil dari perhitungan nilai *round trip time* kemudian akan dibandingkan dan dianalisis untuk menentukan yang mana *access point* yang sah dan yang mana yang merupakan *rogue access point*. Secara teori *rogue access point* yang bertipe *Man-In-The-Middle* akan menghasilkan nilai *round trip time* yang lebih besar dikarenakan ada perbedaan jumlah *hop* yang dilewati. Untuk mengurangi dampak yang ditimbulkan akibat kondisi jaringan yang tidak stabil, digunakan sebuah proses *filtering* dengan metode *k-nearest neighbor* untuk menghilangkan nilai *round trip time* yang anomali.

1.2 Rumusan masalah

1. Bagaimana mekanisme kerja metode perhitungan *rogue access point* dalam melakukan deteksi *rogue access point*?
2. Bagaimana karakteristik nilai *round trip time* yang dihasilkan oleh *rogue access point* dibandingkan dengan *legitimate access point*?
3. Bagaimana tingkat akurasi yang dihasilkan dari pendeteksian *rogue access point*?

1.3 Tujuan

1. Mengetahui mekanisme kerja metode perhitungan *rogue access point* dalam melakukan deteksi *rogue access point*

2. Mengetahui karakteristik nilai *round trip time* yang dihasilkan oleh *rogue access point* dibandingkan dengan *legitimate access point*
3. Mengetahui tingkat akurasi yang dihasilkan dari pendeteksian *rogue access point*

1.4 Manfaat

Manfaat yang diperoleh dari penelitian ini adalah

1. Memperoleh hasil perbandingan perbedaan antara *access point* yang terlegitimasi dan *rogue access point*
2. Dapat mendeteksi *rogue access point* secara akurat sehingga menghindari pengguna terjebak dalam *rogue access point*

1.5 Batasan masalah

1. Metode yang digunakan yaitu perhitungan *round trip time* yang didapatkan dari paket pengujian berupa DNS *lookup*.
2. *Rogue access point* yang difokuskan adalah *rogue access point* yang bertindak sebagai *man in the middle* yang terhubung dengan *access point* yang sah secara *wireless* dan menggunakan *external wireless card* untuk membuat *access point* palsu.
3. Penerapan *rogue access point* dan *PC client* atau pengguna ditempatkan pada lokasi tertentu untuk menghindari *ip conflict*.

1.6 Sistematika pembahasan

Pembuatan tugas akhir ini dilakukan dengan sistematika penulisan sebagai berikut:

BAB I PENDAHULUAN

Berisi latar belakang penulisan pemilihan topik penelitian yang dilakukan oleh penulis, rumusan masalah, tujuan, batasan masalah, manfaat, serta sistematika penulisan skripsi mengenai deteksi *rogue access point* dengan memanfaatkan perbedaan *round trip time*.

BAB II LANDASAN KEPUSTAKAAN

Berisi teori tentang kajian pustaka dari penelitian sebelumnya yang terkait dengan penelitian yang dilakukan penulis serta beberapa teori digunakan untuk menunjang penelitian ini seperti *rogue access point, round trip time*, *k-nearest neighbour*, *wireless security*.

BAB III METODOLOGI PENELITIAN

Berisi metode atau prosedur yang digunakan dalam penelitian ini yang terdiri dari studi literatur mengenai deteksi *rogue access point* dengan perhitungan *round trip time*, rancangan sistem, analisis kebutuhan, rancangan lingkungan

pengujian sistem, implementasi lingkungan pengujian sistem, implementasi sistem, pengumpulan data perbandingan, analisis hasil perbandingan, dan pengambilan kesimpulan.

BAB IV PERANCANGAN DAN IMPLEMENTASI

Berisi prosedur untuk melakukan perancangan dan implementasi dari rancangan ruang lingkup pengujian sistem yang digunakan untuk mendeteksi *rogue access point* dengan melakukan perhitungan *round trip time*. Selain itu juga dijelaskan mengenai prosedur dalam melakukan perbandingan masing-masing *access point*.

BAB V PENGUJIAN DAN ANALISIS

Berisi tentang pengujian sistem sekaligus melakukan perbandingan dari hasil perhitungan *round trip time* tiap *access point* berdasarakan prosedur yang dilakukan pada bab sebelumnya dan analisis terhadap hasil perbandingan yang diperoleh.

BAB VI PENUTUP

Berisi kesimpulan dari penelitian mengenai deteksi *rogue access point* dengan melakukan perhitungan *round trip time*. Serta saran pengembangan penelitian berikutnya oleh penulis.

BAB 2 LANDASAN KEPUSTAKAAN

Pada bab ini dipaparkan mengenai kajian pustaka dan dasar teori terkait dengan penelitian yang dilakukan penulis. Kajian pustaka yang digunakan berdasarkan pada penelitian sebelumnya terkait dengan metode deteksi *rogue access point*. Sedangkan dasar teori yang digunakan untuk mendukung penelitian ini antara lain *wireless security, rogue access point, round trip time* dan *k-nearest neighbour.*

2.1 Kajian pustaka

Tabel 2.1 Penelitian sebelumnya mengenai deteksi RAP

Penelitian	Judul	Parameter
Hao Han, Bo Sheng, Chiu C Tan, Qun Li, Sanglu Lu	*A Timing-Based Scheme For Rogue AP Detection*	Perbandingan Δt yang didapatkan dari selisih waktu DNS dan *probe message* dengan nilai *threshold* tertentu
Somayeh Nikbakhsh, Mazdak Zamani, Azizah Bt Abdul Manaf, Maziar Janbeglou	*A Novel Approach for Rogue Access Point Detection on the Client-Side*	Perbedaan jumlah *hop* dengan melakukan *tracerouting*

Berdasarkan judul skripsi yang akan disusun, penulis menggunakan penelitian yang relevan yang telah dilakukan sebelumnya. Beberapa penelitian yang digunakan sebagai acuan membahas mengenai deteksi *rogue access point* pada sisi *client*.

Penelitian pertama dilakukan oleh Hao Han, Bo Sheng, Chiu C Tan, Qun Li, Sanglu Lu dengan judul "*A Timing-Based Scheme For Rogue AP Detection*". Penelitian ini bertujuan untuk melakukan pendeteksian atas *rogue access point* dengan perhitungan nilai *round trip time* dari tiap *access point*. Penelitian ini dilakukan dengan cara menghitung nilai *round trip time* DNS dan *probe message* pada suatu *access point*. Untuk menentukan *rogue access point* digunakan perbandingan Δt yang didapat dari selisih *round trip time* DNS dengan *probe message* dengan nilai *threshold* yang tertentu. Digunakan metode perhitungan *outlier filter* untuk mengakomodasi ketika jaringan internet cukup berat sehingga menyebabkan nilai *round trip time* akan sangat bervariasi yang membuat kesulitan dalam melakukan analisis dalam menentukan apakah *access point* yang terhubung merupakan *rogue access point* atau bukan. Dalam penelitiannya mereka mampu mendeteksi *rogue access point* dengan tingkat akurasi mencapai hampir 100% pada jaringan yang lancar dan 60% saat jaringan padat (Han, Sheng, Tan, Li dan Lu, 2011).

Penelitian kedua dilakukan oleh Somayeh Nikbakhsh, Mazdak Zamani, Azizah Bt Abdul Manaf, Maziar Janbeglou dengan judul "*A Novel Approach for Rogue*

Access Point Detection on the Client-Side". Penelitian ini membahas tentang deteksi *rogue access point* di sisi *client* tanpa perlu bantuan *administrator* jaringan. Dalam implementasinya penelitian ini mampu mendeteksi *rogue access point* yang bertindak sebagai *man-in-the middle*. Dalam penelitian ini *rogue access point* diimplementasikan dengan menggunakan 2 *wireless card*. *Wireless card internal* sebagai penghubung pada access point yang sah dan eksternal *wireless card* untuk membuat koneksi palsu yang dikonfigurasi identik seperti *access point* yang sah. Penelitain yang dilakukan tersebut juga mampu mendeteksi kemungkinan *rogue access point* dengan tipe *evil twin*. *Access point* yang terdeteksi akan dibandingkan tiap-tiap SSID dan *MAC address* dari *access point*. Ketika SSID dan *MAC address* sama, akan dilakukan perbandingan *IP address*. Ketika ditemukan *IP address* yang sama maka akan dilakukan mekanisme *tracerouting*. Dalam logika jaringan seharusnya tidak akan ada *IP address* yang sama dalam satu rute karena akan menyebabkan *IP conflict*. Ketika *traceroute* menghasilkan nilai yang berbeda maka *user* akan diperingatkan kemungkinan sedang terhubung ke *evil twin access point*. Ketika *IP address* yang terdeteksi berbeda, akan dilakukan perbandingan *network id*. Jika *network id* sama maka *access point* diindikasikan sebagai *access point* yang sah, hal ini dikarenakan kemungkinan *network id* yang sama disebabkan oleh *administrator* jaringan membuat 2 *access point* dengan tujuan *load balancing*. Situasi selanjutnya ketika *network id* berbeda, maka akan dilakukan *traceroute*. Hasil *traceroute* akan dibandingkan, *access point* dengan jumlah *hop* lebih banyak akan dideteksi sebagai *rogue access point.* Jumlah hop lebih banyak mengindikasikan adanya serangan *man in the middle*. *Hop* lebih ini dihasilkan karena serangan *man in the middle* akan membutuhkan *hop* lebih karena setelah menangkap paket pengguna, *attacker* harus menghubungkan ke *access point* yang sah untuk terhubung ke internet(Somayeh, Azizah, Mazdak dan Maziar, 2012).

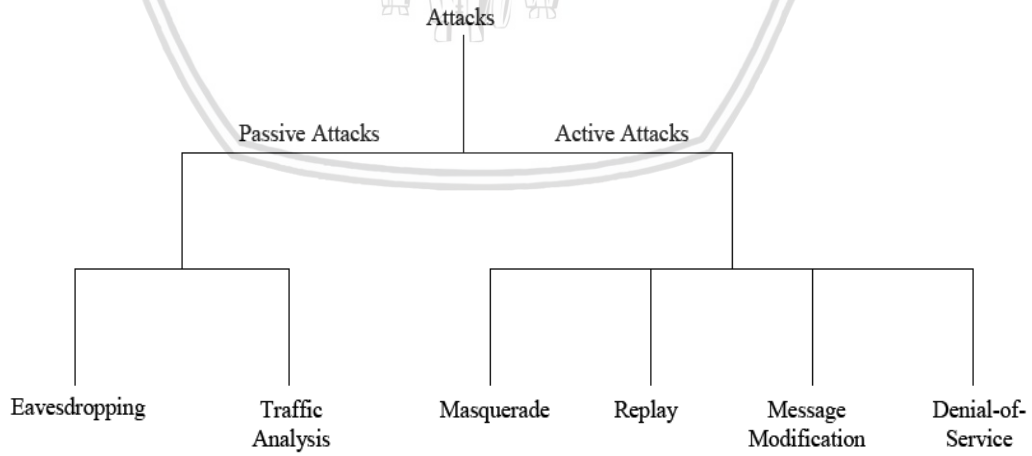
Dalam penelitian yang akan dilakukan oleh penulis akan membahas mengenai deteksi *rogue access point* dengan menghitung *round trip time* pada *access point* yang mempunyai kesamaan SSID dan *MAC address*. *Rogue access point* yang akan dibahas difokuskan pada *rogue access point* yang bertindak sebagai *man in the middle*. *Rogue access point* yang terhubung ke *access point* yang sah, lalu membuat suatu a*ccess point* palsu yang identik dengan yang asli untuk menjebak pengguna. Dengan membuat program yang mampu melakukan perhitungan *round trip time* dan melakukan *filtering* nilai *round trip time* yang didapatkan. Sampai dengan melakukan identifikasi *access point* mana yang dideteksi sebagai *rogue access point*. Sesuai dengan teori *rogue access point* akan menghasilkan nilai *round trip time* yang lebih besar.

2.2 *Wireless security*

Wireless security atau keamanan jaringan nirkabel merupakan pencegahan terhadap akses yang tidak sah atau kerusakan pada suatu computer yang terhubung pada suatu jaringan nirkabel. Standart keamanan jaringan nirkabel dalam IEEE dibagi menjadi 3 dasar, yakni:

1. *Authentication* : Tujuan utama dari WEP adalah menyediakan keamanan untuk dapat melakukan verifikasi identitas dari setiap *client* yang berkomunikasi. Koneksi ke *client* yang tidak terverifikasi keaslian atau keabsahan identitasnya akan ditolak. Layanan ini menjawab pertanyaan "apakah hanya user yang sah yang dapat mengakses jaringan saya?"(Karyiannis dan Owens, 2002).
2. *Confidentiality* : *Confidentiality* dikembangkan untuk menyediakan *privacy* yang sama dengan jaringan dengan kabel. Tujuannya adalah mencegah informasi dari *eavesdropping* atau penyadapan. secara umum *confidentiality* mengakomodasi akan pertanyaan "apakah hanya *user* yang sah yang dapat melihat data saya?"(Karyiannis dan Owens, 2002).
3. *Integrity* : *Integrity* menjamin keamanan bahwa pesan tidak dimodifikasi dalam lalu lintas antara pengguna jaringan nirkabel dan *access point*. *Integrity* menjawab pertanyaan "apakah data datang atau keluar dari jaringan yang terpercaya? Apakah data tidak diubah dalam perjalanannya?"(Karyiannis dan Owens, 2002).

Seringkali makin populernya jaringan nirkabel dimanfaatkan sebagian orang untuk melakukan serangan untuk kepentingan pribadinya. Resiko keamanan jaringan nirkabel relatif lebih rendah dari jaringan dengan kabel. Pada umumnya serangan dalam jaringan nirkabel dibagi menjadi serangan aktif dan pasif. Pada gambar 2.1 dijabarkan bagaimana secara umum *taxonomy* dari serangan pada jaringan nirkabel. Resiko serangan yang lain adalah pengguna jaringan nirkabel tidak bisa memastikan apakah *access point* yang terhubung dengan *gadget* mereka benar-benar *access point* yang sah atau bukan.



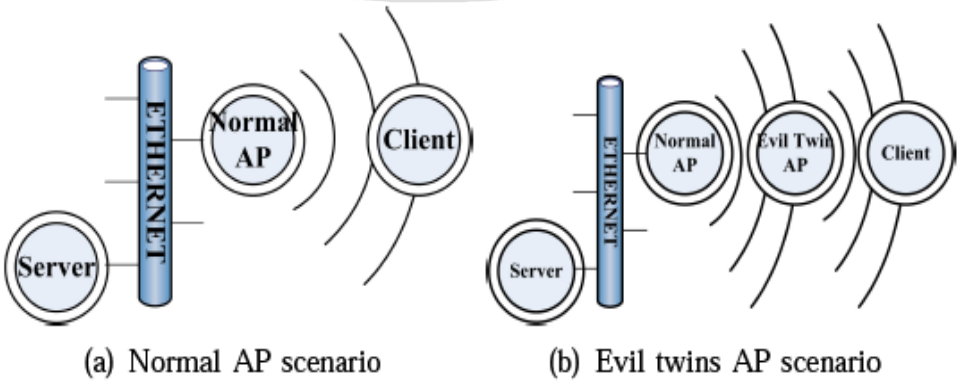
Gambar 2.1 Taksonomi Serangan Keamanan Jaringan Nirkabel

Sumber: Karyiannis dan Owens(2002)

2.3 *Rogue access point*

Rogue access point didefinisikan sebagai *access point* illegal yang tidak dibuat oleh *administrator* jaringan WLAN. *Rogue access point* dapat dibuat dengan dua alat yang berbeda. Yang pertama yaitu dengan menggunakan *wireless router* biasa yang terhubung pada *jack ethernet* untuk membuat koneksi nirkabel yang nantinya akan digunakan untuk menjebak pengguna untuk melakukan koneksi ke *router* tersebut. Yang kedua dengan menggunakan laptop dengan dua buah *wireless card*. *Wireless card* yang pertama digunakan untuk menyambung ke *access point* yang asli, dan yang kedua dikonfigurasikan sebagai *rogue access point* seolah-olah sebagai *access point* yang sah dan menyediakan koneksi internet(Han, Sheng, Tan, Li dan Lu, 2011). Pengguna yang terjebak untuk menyambung ke *rogue access point* akan dengan mudah didapatkan informasi pribadinya.

Rogue access point biasanya dikonfigurasi dengan SSID yang dibuat mirip atau bahkan sama dengan *access point* yang asli. Kebanyakan pengguna yang awam mengira kedua *access point* dengan SSID yang sama merupakan *access point* yang sah. Dan pengguna yang awam kebanyakan memilih *access point* dengan kekuatan sinyal yang lebih tinggi. *Rogue access point* berbeda dengan *roaming wireless network.* Pada *roaming wireless network* seluruh *wifi* memiliki nama yang sama dan konfigurasi keamanan yang sama namun memiliki *mac address* yang berbeda. *Roaming wireless network* bertujuan agar pengguna yang terhubung dapat tersambung ke *access point* terdekat secara otomatis ketika pengguna berpindah-pindah. Walaupun terdiri dari banyak *access point* pada *roaming wireless network,* komputer pengguna hanya akan mendapati sebuah *SSID* ketika melakukan pencarian menggunakan *GUI*. *Rogue access point* yang dikonfigurasi mirip atau sama dengan *access point* yang asli disebut dengan *evil twin access point*. *Evil twin* meniru SSID dan *MAC address* dengan menggunakan cara *mac spoofing* untuk mengubah *MAC address* guna membuat pengguna bingung dalam memilih *access point*. *Evil twin access point* diilustrasikan pada gambar 2.2.



Gambar 2.2 Ilustrasi *Evil Twin Access Point*

Sumber : Song, Yang dan Gu(2010).

Attacker biasanya melancarkan serangan *evil twin* di tempat yang menyediakan *hotspot* gratis, misalnya *airport*, hotel, maupun perpustakaan. Dengan melakukan serangan *evil twin*, *attacker* dapat mengambil data penting semisal *password* atau informasi kartu kredit dengan menyadap *link* komunikasi atau melancarkan serangan *man in the middle*. *Evil twin* merupakan ancaman serius dalam keamanan *wireless LAN*(Song, Yang dan Gu, 2010).

2.4 *Round trip time*

Round trip time didefinisikan sebagai waktu yang dibutuhkan ketika *client* mengirim sebuah *request* sampai *server* memberikan *response* kembali ke *client*. Untuk menghitung *round trip time* dapat dilakukan dengan mengirim pesan *ping*. Selain menggunakan *ping*, untuk mendapatkan waktu *round trip time* juga bisa dihasilkan dari pengiriman paket lain seperti *DNS* maupun *TCP*. Nilai *round trip time* dapat dipengaruhi oleh besarnya paket *request* yang dikirimkan oleh *client* dan kondisi trafik jaringan yang berubah-ubah. Ketika kondisi trafik jaringan relatif lancar, nilai *round trip time* akan kecil dan ketika kondisi jaringan lambat nilai *round trip time* yang dihasilkan akan besar. Nilai *round trip time* dapat dihitung secara manual dari hasil tangkapan menggunakan *wireshark* maupun menggunakan *tool* seperti *dig*. *Query time* yang dihasilkan *dig* merupakan nilai *round trip time* dari paket DNS *lookup*

Round trip time sering digunakan untuk menentukan kondisi trafik jaringan atau bagus tidaknya kondisi jaringan saat itu. Informasi yang dihasilkan oleh *round trip time* dapat digunakan untuk melakukan analisa mengenai penyebab terjadinya kemacetan lalu lintas pengiriman data. *Round trip time* juga dapat digunakan untuk membantu memanajemen *queue* dan penyediaan *buffer* yang efisien.

2.5 DNS

DNS merupakan singkatan dari *domain name system.* Pada dasarnya fungsi *DNS* yaitu memetakan alamat *website* yang mudah dibaca pengguna ke alamat IP. Biasanya dalam suatu jaringan menyimpan *DNS* secara lokal didalam *cache* dengan tujuan meningkatkan performansi. *DNS lookup query* dibagi menjadi 2 macam yaitu, *recursive* dan *non-recursive*. Dalam *DNS lookup recursive*, user mengirim *query* untuk meminta pada *server* lokal sebuah *hostname*. Jika *server* lokal tidak bisa menjawab *query* maka akan mengontak *root DNS server* yang kemudian akan menanyakan pada *server* lain untuk menentukan *IP address*. Sementara *non-recursive DNS* hanya akan mencari *query* yang diminta pada *DNS server* lokal yang tersimpan di *cache*. Jika tidak tersedia maka akan ditampilkan pesan bahwa *query* tidak ditemukan (Han, Sheng, Tan, Li dan Lu, 2011).

Untuk menjalankan *query DNS lookup* bisa dilakukan melalui *tools* maupun *command line*. Dengan menggunakan *command line* user bisa melakukan eksekusi perintah *nslookup* untuk mengirim *query DNS*. Dari eksekusi *nslookup* waktu *request-response DNS* bisa dilihat dari hasil tangkapan melalui *wireshark*. *Query DNS* juga bisa didapatkan dengan mengeksekusi perintah *dig*. *Query time*

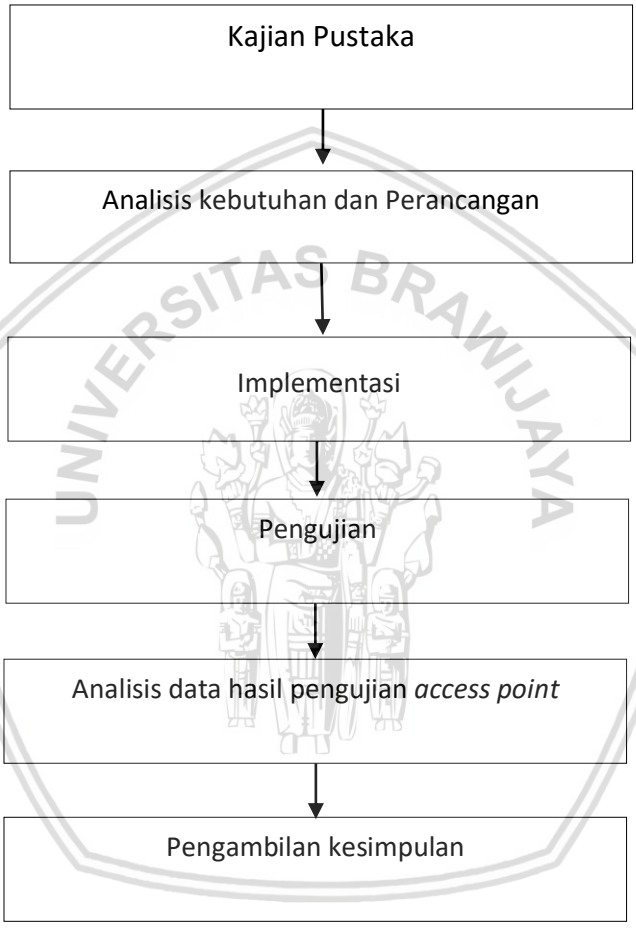
yang didapatkan dari perintah *dig* merupakan angka *round trip time* atau waktu *request-response* dari DNS *lookup*. Sementara untuk *tools* lain untuk mengirim *DNS lookup* bisa menggunakan *DNS lookup* dari *nirsoft*.

2.6 *K-nearest neighbor*

K-Nearest Neighbor (KNN) adalah suatu metode untuk melakukan klasifikasi berdasarkan kedekatan jarak suatu data dengan data yang lain. KNN merupakan salah satu algoritma yang paling banyak digunakan dalam *machine learning*. KNN adalah algoritma *supervised learning* dengan kata lain hasil dari *query instance* yang baru akan diklasifikan berdasarkan mayoritas dari kategori pada KNN. Pengklasifikasian dalam KNN dapat digunakan untuk melakukan *filtering outlier*. Yakni melakukan penyaringan nilai yang unik atau berbeda pada suatu data dengan parameter tertentu. Dalam melakukan *filtering* KNN akan mencari jarak terdekat suatu obyek pada obyek lain. Jarak terjauh yang dihasilkan dapat dikenali sebagai *outlier*. Nilai *outlier* merupakan nilai anomali yang dapat dihapus dengan jumlah *threshold* tertentu.

BAB 3 METODOLOGI

Dalam bab ini dijelaskan mengenai prosedur yang digunakan untuk melakukan deteksi *rogue access point* di sisi *client* dengan metode perhitungan *round trip time*. Prosedur yang digunakan sesuai dengan alur diagram pada **Gambar 3.1**.



Gambar 3.1 Alur diagram metode penelitian

Untuk melakukan deteksi *rogue access point* terlebih dahulu akan ditentukan teknik dan langkah-langkah dalam melakukan deteksi *rogue access point*. Deteksi *rogue access point* dilakukan dengan cara melakukan *scanning* pada seluruh *access point* yang ada pada suatu area. Kemudian melakukan identifikasi SSID *access point*. Jika terdapat 2 atau lebih *access point* dengan SSID yang sama, kemungkinan salah satu dari *access point* yang terdeteksi adalah *rogue access point* yang melakukan SSID *masquerading* untuk membuat seolah-olah kedua *access point* adalah sah. Peneliti akan melakukan sambungan ke

kedua *access poin*t dengan SSID yang sama secara bergantian dan membandingkan *IP address* kedua *access point* tersebut. Dengan estimasi bahwa *rogue access point* memanfaatkan *access point* yang sah untuk memberikan koneksi yang palsu maka seharusnya nilai *round trip time* dari kedua *access point* akan berbeda. *Rogue access point* akan memiliki nilai *round trip time* yang lebih besar dikarenakan jumlah *hop* dari *rogue access point* akan lebih banyak dari *access point* yang sah. Program yang dibuat akan menjalankan prosedur dengan melakukan sambungan ke kedua *access point* dan menghitung nilai *round trip time* masing-masing dan kemudian membandingkannya. Paket yang dikirim sebagai paket uji untuk mendapatkan nilai *round trip time* yakni berupa paket *dns lookup*.

3.1 Studi literatur

Studi literatur membahas mengenai dasar teori yang digunakan dalam penelitan dan penulisan tugas akhir ini. Dasar teori diperoleh dari penelitian relevan yang dilakukan sebelumnya dan terkait dengan penelitian yang dilakukan oleh penulis, seperti dari jurnal, *e-book*, dan situs *web* resmi yang dapat dipertanggung jawabkan. Dasar teori yang digunakan untuk menunjang penelitian ini antara lain *rogue access point, round trip time*, *wireless security* dan *python*.

3.2 Perancangan dan analisis kebutuhan deteksi *rogue access point*

Tahapan ini membahas tentang rancangan implementasi deteksi *rogue access point* langkah-langkah yang dilakukan untuk mengukur nilai *round trip time* kedua access point untuk kemudian akan membandingkan nilai *round trip time* dan menganalisa dari hasil perbandingan. Dalam tahapan ini juga dibahas mengenai analisis kebutuhan yang digunakan untuk menunjang penelitian yang dilakukan.

3.2.1 Analisis Kebutuhan sistem

Dalam tahapan ini dijelaskan mengenai analisis kebutuhan dari sistem pendeteksian *rogue access point* dengan metode perhitungan *round trip time*. Kebutuhan sistem meliputi kebutuhan fungsional sistem sampai dengan kebutuhan lingkungan implementasi sistem. Analisa kebutuhan sistem bertujuan untuk mempermudah pembuatan sistem pendeteksian *rogue access point* dengan metode perhitungan *round trip time*. Dan juga membatasi sistem yang dibuat agar tidak terlalu melebar sampai mencakup hal-hal yang sebenarnya tidak diperlukan oleh sistem.

3.2.1.1 Kebutuhan fungsional

Kebutuhan fungsional merupakan kebutuhan utama atau segala fungsi utama yang diperlukan dalam implementasi sistem. Dalam implementasi sistem pendeteksian *rogue access point* dengan metode perhitungan *round trip time*,

kebutuhan utama yang harus dapat dilakukan sistem adalah melakukan perhitungan *round trip time*. Kebutuhan fungsional sistem sebagai berikut :

1. Sistem mampu membaca daftar dalam file teks.
2. Sistem mampu mengirimkan pesan DNS *lookup*
3. Sistem mampu menghitung nilai *round trip time*
4. Sistem mampu melakukan *filtering* nilai *round trip time*
5. Sistem mampu membandingkan nilai *round trip time* yang didapatkan dari kedua *access point access point* yang diuji
6. Sistem mampu menyimpan data nilai *round trip time* yang telah didapatkan.

3.2.1.2 Kebutuhan lingkungan sistem

Kebutuhan yang digunakan dalam pengujian maupun implementasi deteksi *rogue access point* adalah :

1. Kebutuhan Perangkat keras.

Dalam implementasinya perangkat keras yang dibutuhkan antara lain :

1. *Access point*. *Access point* digunakan sebagai *legitimate access point* yang menyediakan layanan internet yang *legal*.
2. 1 Laptop dengan 2 buah *wireless card* sebagai *rogue access point*. Satu *wireless card* akan menyambung ke *legitimate access point* guna mendapatkan koneksi internet. Sementara satu *wireless card* yang lain akan bertindak seolah-olah *access point* yang asli dengan membuat konfigurasi yang sama dengan *access point* yang asli.
3. 1 buah laptop untuk melakukan pengujian untuk melakukan sambungan ke kedua *access point* dan membandingkan hasil perhitungan *round trip time*.

2. Kebutuhan Perangkat Lunak

Kebutuhan akan perangkat lunak yang dibutuhkan antara lain :

1. *Software access point* untuk membuat *access point* yang akan dikonfigurasi sebagai *rogue access point*. Dalam penelitian ini akan digunakan airmon yang disediakan oleh *software* aircrack karena *tool* ini mudah digunakan, mempunyai banyak literature dan tutorial yang mendukung dan mendukung banyak fungsi untuk melakukan serangan.
2. *Python* digunakan sebagai bahasa pemrograman untuk membuat program yang mampu melakukan sambungan ke kedua *access point* yang dicurigai, mengirim paket *dns lookup*, melakukan perhitungan *round trip*

time dan sekaligus melakukan perbandingan nilai *round trip time* yang telah didapat.

3.2.2 Perancangan sistem pendeteksian *rogue access point*

Pada tahap ini dijelaskan mengenai deskripsi dan alur kerja program yang akan dibuat. Deskripsi program nantinya akan sesuai dengan implementasi program pendeteksian *rogue access point* dengan metode perhitungan *round trip time* yang akan dibuat. Alur kerja program menjelaskan bagaimana sistem pendeteksian *rogue access point* dengan metode perhitungan *round trip time* akan berjalan. Mulai dari ketika dicurigai adanya *access point* dengan SSID yang sama yang menandakan adanya potensi keberadaan *rogue access point*, hingga melakukan pendeteksian dengan menentukan *access point* yang dianggap sebagai *rogue access point*. Alur kerja program lebih lengkap akan digambarkan dalam sebuah *flowchart* yang menggambarkan tiap proses yang dilakukan oleh program.

3.3 Implementasi sistem pendeteksian *rogue access point*

Pada tahap ini dijelaskan mengenai proses implementasi sistem pendeteksian *rogue access point*. Langkah-langkah implementasi yang akan dilakukan sebagai berikut :

1. Implementasi proses membaca domain dari *file* teks
2. Implementasi proses pengiriman DNS *lookup*
3. Implementasi perhitungan *round trip time*
4. Implementasi proses *filtering*
5. Implementasi proses perbandingan RTT dan deteksi
6. Implementasi proses penyimpanan data nilai RTT

3.4 Pengujian sistem pendeteksi *rogue access point*

Pengujian dilakukan untuk menghitung seberapa besar akurasi pendeteksian sistem dan juga seberapa baik *filtering* mampu mengurangi dampak dari kondisi jaringan yang tidak stabil pada tingkat akurasi pendeteksian *rogue access point*

3.4.1 Pengujian perhitungan *round trip time*

Pengujian *rogue access point* akan dilakukan ketika mendapati dua buah *access point* dengan SSID sama. Program akan dijalankan hingga program mendapatkan nilai *round trip time* dari kedua *access point* yang dicurigai. Nilai *round trip time* dari kedua *access point* tersebut akan menjadi parameter untuk menentukan *access point* mana yang dideteksi sebagai *rogue access point*.

Sebagai perbandingannya dalam melakukan *DNS Non-recursive query* pada *legitimate access point* jalur yang ditempuh paket adalah sebagai berikut :

STATION → AP → DNS SERVER → AP → STATION

Sementara pada *rogue access point* ketika melakukan *query DNS* secara *non-recursive* jalur yang ditempuh akan bertambah karena akan tetap melewati *access point* yang sah yakni :

STATION → RAP → AP → DNS SERVER → AP → RAP → STATION

Pesan yang digunakan adalah *DNS lookup* karena terkadang jika menggunakan *ping* sebagai paket penguji, *rogue access point* yang cerdas bisa langsung mengembalikan tanpa harus mengontak *access point* yang asli. Hal ini mengakibatkan nilai dari *round trip time* dari *rogue access point* bisa jadi akan lebih kecil. Perhitungan *round trip time* dari *access point* akan dilakukan dengan menggunakan *non-recursive DNS query*. Hal ini dikarenakan fungsi *DNS lookup* merupakan kewajiban yang harus dimiliki tiap jaringan. Hal ini memaksa *rogue access point* untuk meneruskan *query* secara *recursive* ke *legitimate access point*. Hal ini kemudian akan menimbulkan penambahan jumlah *hop* yang menghasilkan *delay* yang tidak bisa dihindari dan membuat waktu dari *round trip time* total akan bertambah. *Query Non recursive DNS* akan dilakukan n-kali untuk kemudian akan dilakukan rata-rata. Rata-rata *round trip time* dari *DNS lookup* antara kedua *access point* yang identik inilah yang nantinya akan dibandingkan.

3.4.2 Pengujian proses *filtering*

Dalam suatu kasus yang disebabkan karena keadaan jaringan yang dinamis, maka akan dibuat *filtering outlier*. *Filtering outlier* dibuat untuk mengatasi permasalahan kondisi jaringan yang tiba-tiba lambat dan mengakibatkan nilai *round trip time* yang variatif. Dalam suatu kasus misalnya dengan 50 sample *round trip time*, 49 diantaranya menghasilkan nilai *round trip time* 1 ms sedangkan 1 buah *round trip time* menghasilkan nilai 100 ms. Hal ini mengakibatkan nilai rata-rata menjadi 2,98 ms dan hasil tersebut tidak menunjukkan mayoritas hasil pengujian. Dan nilai 100 ms nantinya akan dianggap anomali dan dilakukan *filtering* dengan menggunakan metode *k-nearest neighbour* sehingga hasilnya akan lebih akurat. Pengujian fungsi *filtering* akan dilakukan dengan menggunakan *tool network shapper* untuk menambah *delay* paket secara sengaja atau melakukan simulasi ketika jaringan atau trafik sedang lambat.

3.5 Analisa data hasil perbandingan *access point*

Data yang diperoleh dari perhitungan nilai *round trip time* dari program yang dijalankan akan dianalisa. Secara teori nilai *round trip time* dari *rogue access point* akan selalu lebih besar karena perbedaan jumlah *hop* yang dilalui. Analisa sistem pendeteksi *rogue access point* akan menjelaskan mengenai hasil akurasi dari sistem pendeteksian *rogue access point* yang dilakukan. Akan dilakukan analisa mengenai seberapa banyak pengaruh kondisi jaringan yang tidak stabil akan mengurangi akurasi pendeteksian. Analisa pendeteksian *rogue access point* pada jaringan yang berbeda untuk memberikan perbandingan nilai akurasi jika program dijalanakan pada beberapa jaringan yang berbeda.

Pengaruh kondisi trafik jaringan yang tidak stabil akan diminimalisir dengan menggunakan fungsi *filtering* untuk menyaring nilai *round trip time* yang anomali. Data yang dihasilkan dari program yang diambil adalah rata-rata nilai *round trip time* setelah dilakukan *filtering* dan sebelum dilakukan *filtering.* Hal ini untuk menguji seberapa efektif penggunaan fungsi *filtering* untuk meningkatkan akurasi deteksi *rogue access point* secara keseluruhan. Hasil yang didapatkan dari *filtering* akan dibandingkan dengan eksekusi program tanpa proses *filtering*. Hal ini untuk mengetahui seberapa besar proses *filtering* mampu meningkatkan akurasi pendeteksian *rogue access point*.

3.6 Pengambilan kesimpulan dan saran

Setelah melakukan analisis dari hasil perbandingan *round trip time access point*, peneliti akan menarik kesimpulan. Kesimpulan dibuat untuk menjawab permasalahan yang telah dirumuskan sebelumnya yang diperoleh dari analisis hasil pengujian. Kesimpulan ditulis ketika seluruh tahapan penelitian telah selesai dibuat. Saran dituliskan untuk memberikan informasi yang diharapkan dapat digunakan sebagai acuan untuk penelitian selanjutnya sehingga kekurangan yang dihasilkan pada penelitian ini dapat diperbaiki.

BAB 4 ANALISIS KEBUTUHAN DAN PERANCANGAN

Dalam bab ini dijelaskan mengenai prosedur yang terdiri dari proses analisis kebutuhan sistem dan perancangan sistem. Analisa kebutuhan meliputi segala sesuatu yang harus bisa dilakukan oleh sistem sampai dengan kebutuhan implementasi sistem. Pada subbab perancangan berisi mengenai deskripsi sistem, bagaimana sistem akan mengakomodasi fungsi-fungsi utama yang dijabarkan dalam analisis kebutuhan dan cara kerja sistem dalam melakukan deteksi *rogue access point*.

4.1 Analisis kebutuhan sistem

Analisis kebutuhan sistem dilakukan untuk menentukan kebutuhan apa saja yang diperlukan dalam pembuatan sistem pendeteksian *rogue access point* dengan metode perhitungan *round trip time* ini. Analisis kebutuhan sistem bertujuan untuk mempermudah implementasi agar proses implementasi tidak melebar maupun tidak mengurangi kebutuhan-kebutuhan utama dari sistem pendeteksian *rogue access point* dengan metode perhitungan *round trip time*. Dalam analisis kebutuhan sistem pendeteksian *rogue access point* dengan metode perhitungan *round trip time* ini akan dijabarkan mengenai kebutuhan fungsional dari sistem dan kebutuhan lingkungan implementasi sistem.

4.1.1 Kebutuhan fungsional

Kebutuhan fungsional meliputi fungsi-fungsi utama yang harus ada dalam sistem pendeteksian *rogue access point* dengan metode perhitungan *round trip time*. Kebutuhan fungsional yang dirancang harus ada dalam program yang nantinya akan diimplementasikan. Untuk sistem pendeteksian *rogue access point* dengan metode perhitungan *round trip time*, kebutuhan fungsionalnya meliputi :

1. Sistem mampu membaca *domain list* yang disimpan dalam file teks.
2. Sistem mampu mengirimkan pesan DNS *lookup* ke tiap-tiap *domain*
3. Sistem mampu menghitung nilai *round trip time* yang didapat dari pengiriman pesan DNS *lookup*.
4. Sistem mampu melakukan *filtering* nilai *round trip time* yang tiba-tiba meninggi atau anomali.
5. Sistem mampu membandingkan nilai *round trip time* yang didapatkan dari kedua *access point access point* yang diuji dan mendeteksi *rogue access point*.
6. Sistem mampu menyimpan data nilai *round trip time* yang telah didapatkan.

4.1.2 Kebutuhan lingkungan implementasi sistem

Dalam pembuatan sistem pendeteksian *rogue access point* dengan metode perhitungan *round trip time* diperlukan lingkungan implementasi untuk mendukung proses implementasi. Lingkungan implementasi meliputi kebutuhan akan *hardware* dan *software* yang digunakan untuk mengimplementasikan sistem pendeteksian *rogue access point* dengan metode perhitungan *round trip time*.

Sistem akan dijalankan pada sebuah komputer *client* yang tersambung kepada kedua *access point* dan melakukan perhitungan *round trip time* kepada kedua *access point*. Komputer *client* akan berjalan pada sistem operasi linux ubuntu. Dengan menggunakan bahasa pemrograman python untuk membuat program implementasi sistem pendeteksian *rogue access point* dengan metode perhitungan *round trip time.* Komputer *client* akan diinstall *software visual studio code* sebagai program untuk *editor* kode program.

4.2 Perancangan sistem pendeteksian *rogue access point*

Sistem pendeteksian *rogue access point* akan dirancang sesuai dengan kebutuhan sistem yang telah dijabarkan pada subbab analisa kebutuhan sistem sebelumnya. Sistem harus mampu mengakomodasi tiap fitur atau fungsi yang telah disusun. Perancangan sistem pendeteksian *rogue access point* meliputi deskripsi sistem dan alur kerja sistem yang digambarkan dengan menggunakan *flowchart*.

4.2.1 Deskripsi sistem

Sistem pendeteksian *rogue access point* dengan metode perhitungan *round trip time* akan melakukan perhitungan nilai *round trip time* dengan menggunakan paket uji berupa DNS *lookup* yang dikirim oleh program yang dijalankan pada komputer *client*. Program akan mengambil 50 *sample domain* yang dibaca dari file teks yang berisi daftar *domain* yang didapatkan dari *history browser*. Ke 50 *domain* ini dipastikan benar-benar *valid* untuk menghindari kesalahan pendeteksian.

Untuk pengiriman DNS *lookup* akan digunakan fitur *dig* pada *command line* linux yang biasa digunakan untuk mendapatkan *query time* dari DNS *lookup*. *Dig* digunakan karena hasil *query time* yang didapatkan dari fungsi *dig* sama dengan nilai *round trip time* DNS *lookup* yang didapat dari *capture* wireshark yang dihitung secara manual. *Query time* atau nilai *round trip time* yang didapat dari setiap *domain* dari ke 50 *domain* yang ada pada file teks akan dilakukan perhitungan rata-rata.

Dari rata-rata *round trip time* yang didapatkan akan dilakukan *filtering* untuk menghapus nilai *round trip time* yang anomali. Nilai anomali ini akan menyebabkan kemungkinan kesalahan dalam pendeteksian *rogue access point*. Sebagai contoh dalam 50 *domain* yang diuji dalam suatu waktu terjadi gangguan koneksi internet yang menyebabkan satu nilai *round trip time* menjadi tinggi.

Filtering digunakan untuk menghilangkan nilai yang anomali tersebut. Metode yang digunakan untuk melakukan proses *filtering* yakni *k-nearest neighbor*.

Algoritma yang digunakan adalah sebagai berikut :

Algoritma *K-nearest neighbors untuk filtering outlier*

1: k = m = 0.2 x n

2: for i = 1 to n do

3: $d_i = d^k (RTT_i)$

4: end for

5: Sort all di in increasing order

6: Remove the top m largest values

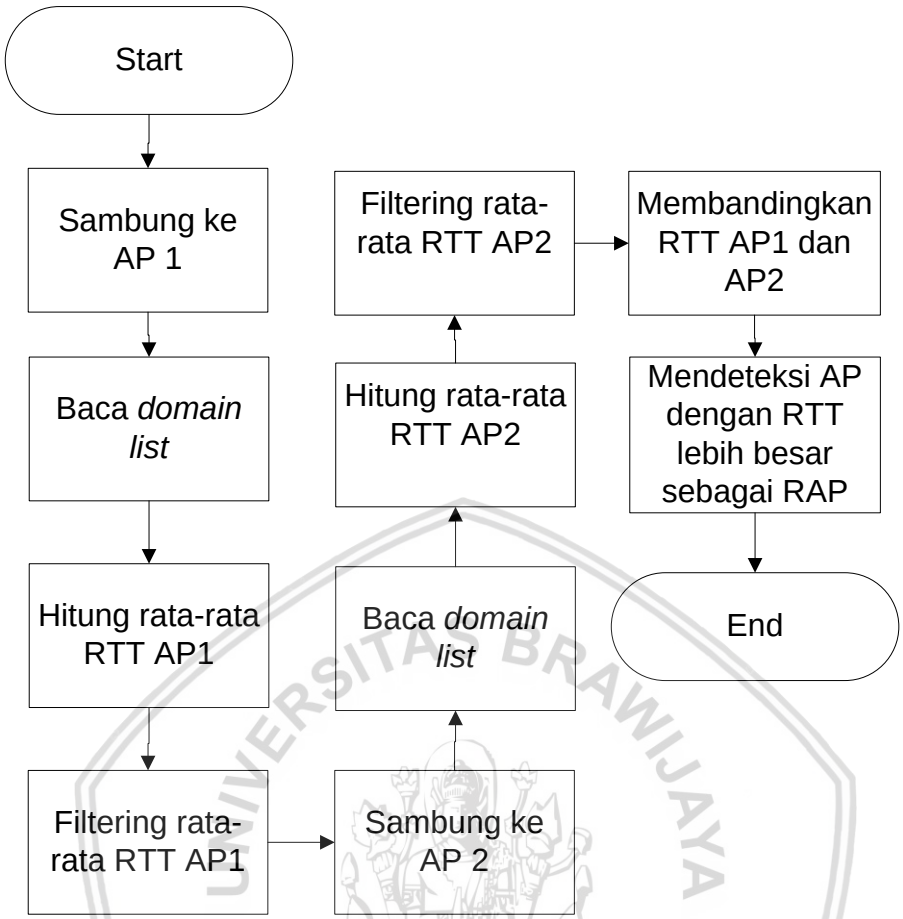
d^k merupakan jarak antara rata-rata awal dengan nilai *round trip time*. *Filtering* diatur sebanyak 20% dari *n* atau jumlah data. Nilai d^k akan diurutkan dengan nilai d^k terbesar sebanyak *m* akan dihapus

Dari 50 nilai *round trip time* dari tiap domain akan dihitung rata-rata awal. Nilai *round trip time* dari tiap *domain* akan dihitung jarak dengan rata-rata awal yang didapatkan sebelumnya. 20 persen data nilai *round trip time* dengan jarak terjauh akan dihapus. Kemudian akan dihitung kembali rata-rata yang baru dari data yang tersisa.

Program melakukan seluruh perhitungan pada kedua *access point* dan melakukan perbandingan hasil rata-rata nilai *round trip time* yang didapatkan. Nilai rata-rata *round trip time* yang didapatkan akan disimpan dalam file teks. Program mendeteksi *access point* dengan rata-rata nilai *round trip time* lebih tinggi sebagai *rogue access point*.

4.2.2 Alur kerja sistem

Alur kerja atau jalannya sistem pendeteksian *rogue access point* dengan metode perhitungan *round trip time* digambarkan melalui *flowchart* yang diperlihatkan pada **gambar 4.1.**



Gambar 4.1 *Flowchart* alur kerja program

Pada **gambar 4.1** diperlihatkan urutan alur kerja program. Program akan berjalan dengan melakukan sambungan ke salah satu *access point* yang dicurigai sebagai *rogue access point*. Setelah mendapatkan sambungan, program melakukan perhitungan nilai *round trip time* dari 50 *domain* yang didapat dari sebuah file teks dengan mengirim DNS *lookup*. Program akan melakukan perhitungan rata-rata nilai *round trip time* lalu melakukan proses *filtering*. Dalam proses *filtering* akan melakukan penyaringan nilai yang anomali agar didapat nilai rata-rata akhir yang lebih baik. Selanjutnya program melakukan sambungan ke *access point* kedua dan melakukan proses yang sama seperti yang dilakukan pada *access point* pertama.

Dari nilai rata-rata akhir *round trip time* yang didapatkan dari perhitungan yang dilakukan akan dilakukan perbandingan. Nilai rata-rata *round trip time* dari *access point* satu dan *access point* 2 akan dibandingkan. *Access point* dengan nilai *round trip time* lebih tinggi akan dideteksi sebagai *rogue access point*. Nilai *round trip time* yang lebih tinggi ini dihasilkan oleh perbedaan jumlah *hop* yang dilalui oleh paket.

BAB 5 IMPLEMENTASI

Langkah implementasi pada penelitian ini terdiri dari implementasi perhitungan *round trip time* dan implementasi metode *filtering*. Implementasi sistem pendeteksian *rogue access point* akan dilakukan dengan membuat aplikasi yang dapat melakukan perhitungan nilai *round trip time* dari DNS *lookup*, melakukan *filtering* rata-rata *round trip* time, sekaligus membandingkan hasil dari kedua perhitungan pada *acces point* yang identik. Aplikasi ini akan diimplementasikan menggunakan bahasa pemrograman *python*.

5.1 Implementasi proses membaca *domain* dari file teks

Domain website digunakan sebagai tujuan pengiriman paket uji berupa DNS *lookup*. Daftar *domain website* disimpan dalam sebuah file teks bernama *test.txt*. File ini berisi 50 *domain* yang akan digunakan untuk mendapatkan nilai *round trip time*. Keseluruhan *domain* yang ada pada file *test.txt* telah diuji sebelumnya untuk memastikan *domain* benar-benar valid dengan cara dilakukan akses dari *browser* ke setiap *domain* yang ada pada daftar. Pada **gambar 5.1** diperlihatkan implementasi proses pembacaan *domain* yang ada pada file *test.txt* kedalam kode program.

```
with open("test.txt", "r") as ins:
    for line in ins:
```

Gambar 5.1 Kode program proses membaca domain dari file teks

5.2 Implementasi proses pengiriman DNS *lookup*

DNS *lookup* digunakan sebagai paket pengujian yang dikirim untuk mendapatkan nilai *round trip time*. Program akan melakukan *query* DNS *lookup* dengan menggunakan *command dig* yang ada pada *terminal linux*. Pada program perintah *dig* ini dituliskan dalam *subproccess*. Pada **gambar 5.2** diperlihatkan implementasi proses pengiriman DNS *lookup* dengan menggunakan perintah *dig* ke dalam kode program. Fungsi *subproccess dig* dapat dipanggil pada program utama. *Query time* yang didapat dari perintah *dig* merupakan representasi dari nilai *round trip time*. Perintah *dig* dijalankan pada seluruh *domain* yang ada didalam daftar pada file *test.txt*.

```
def get_query_time(url):
    prepare = "dig {0} | grep \"Query time\"".format(url)
    awkstr = " | awk '{print $4}'"
    command = prepare + awkstr
    process = subprocess.Popen(command,
    shell=True,
    stdout=subprocess.PIPE)
    line = process.stdout.readline()
    return line.rstrip().decode("utf-8")
```

Gambar 5.2 *Subproccess* pengiriman DNS *lookup*

5.3 Implementasi perhitungan *round trip time*

Perhitungan *round trip time* merupakan fitur utama dari program sistem pendeteksian *rogue access point* ini. Untuk melakukan perhitungan *round trip time* juga melibatkan fitur-fitur lain. Pada **gambar 5.3** diperlihatkan baris kode program yang digunakan untuk melakukan perhitungan nilai *round trip time.*

```
with open("test.txt", "r") as ins:
    for line in ins:
        rtt = float(bigmac.get_query_time(line.strip()))
        jumlah += rtt
        arr_rtt.append(rtt)
        print('DNS time  = ', arr_rtt[jumlahdata])
        print(line.strip())
        jumlahdata += 1

print(jumlah)
print(jumlahdata)
first_avg = jumlah / float(jumlahdata)
print('Rata-rata Awal = ', first_avg)
```

Gambar 5.3 Baris kode program perhitungan nilai *round trip time*

Baris kode program pada **gambar 5.3** merupakan implementasi perhitungan *round trip time* yang diterapkan pada bahasa pemrograman *python*. Baris program *"bigmac.get_query_time"* merupakan fungsi untuk memanggil fungsi DNS *lookup* yang didefinisikan dalam *subprocess.* Nilai *round trip time round trip time* yang didapat dari pengiriman DNS *lookup* dengan menggunakan perintah *dig* pada setiap *domain* dalam *test.txt* akan disimpan dalam *array*. Seluruh nilai *round trip time* dari semua *domain* yang diuji akan dijumlahkan dan dibagi dengan jumlah *domain* untuk mendapatkan nilai rata-rata awal.

Untuk menghindari penyimpanan DNS *cache* di sisi computer *client*, dibuat fungsi untuk melakukan penghapusan atau *flush* DNS *cache*. Fungsi ini didapatkan dengan menggunakan perintah yang dipanggil dari *command line* yakni :

"system-resolve –flush-cache"

Dalam program fungsi ini dituliskan dengan menggunakan *subprocess* seperti yang diperlihatkan pada **gambar 5.4**. Fungsi ini dipanggil setiap program selesai melakukan satu kali perhitungan *round trip time* dari masing-masing *access point* yang dicurigai. Adanya penyimpanan DNS *cache* di sisi *client* akan membuat nilai *round trip time* yang didapatkan menjadi bernilai 0 ms. Sehingga akan sangat berpengaruh pada pendeteksian.

```
def flush_dns_cache():
    process = subprocess.Popen("systemd-resolve --flush-cache",
    shell=True,
    stdout=subprocess.PIPE)
    line = process.stdout.readline()
    return line.rstrip().decode("utf-8")
```

Gambar 5.4 *Subproccess flushing* DNS *cache*

5.4 Implementasi proses *filtering*

Untuk mengatasi kondisi jaringan yang seringkali tiba-tiba lambat atau trafik jaringan yang tiba-tiba meninggi, maka dibutuhkan metode untuk menyaring nilai *round trip time* dari *DNS* ketika jaringan mengalami hal tersebut. Sebagai contoh ketika melakukan pengujian terhadap *sample* 50 nilai *round trip time* dari 49 *sample* didapatkan angka 1 ms sedangkan satu nilai sebesar 100 ms. Nilai 100 ms ini kemungkinan terjadi ketika jaringan sedang mengalami kondisi dimana trafik sedang sibuk atau tinggi sehingga mempengaruhi nilai rata-rata dari *round trip time* yang sebenarnya.

Untuk mengatasi hal ini dibuatlah metode *filtering* untuk menghilangkan nilai yang dideteksi sebagai anomali ini. Metode yang digunakan yakni *KNN* satu dimensi. Metode ini dilakukan dengan mengambil rata-rata dari 50 nilai *round trip time* yang ada. Kemudian setiap nilai *round trip time,* akan dicari jarak dengan nilai rata-rata awal tersebut. Nilai yang terbesar atau dengan kata lain mempunyai jarak terjauh akan dideteksi sebagai anomali. Dari data jarak masing-masing *round trip time* dengan rata-rata awal akan disaring 20 persen dari jarak yang terjauh. Sehingga jika dari 50 data 10 data dengan jarak terjauh akan dihilangkan. Kemudian data yang telah disaring akan kembali dirata-rata untuk mendapatkan nilai rata-rata baru yang lebih baik.

```
32  arr_distance = []
33  for r in range(len(arr_rtt)):
34      jarak = abs(first_avg - arr_rtt[r])
35      arr_distance.append(jarak)
36  print(arr_distance)
37  print("------ FILTERED DATA ------")
38  sorted_distance = sorted(range(len(arr_distance)), key=lambda k: arr_distance[k])
39  print(sorted_distance)
40  jumlahdata = len(sorted_distance) - int(0.2 * jumlahdata)
41  jumlahbaru = 0
42  for f in range(jumlahdata):
43      jumlahbaru += arr_rtt[sorted_distance[f]]
44  print(jumlahbaru)
45  rata_rata = jumlahbaru / float(jumlahdata)
46  print('Rata-rata = ', (rata_rata))
```

Gambar 5.5 Implementasi *filtering* pada *python*

Pada **gambar 5.5** diperlihatkan bagaimana proses *filtering* diimplementasikan ke dalam kode program dengan bahasa pemrograman *python*. Nilai dari rata-rata awal yang didapatkan dari perhitungan *round trip time* akan dikurangkan dengan nilai *round trip time* dari tiap *domain* yang diuji. Hasil pengurangan akan disimpan sebagai jarak. Program melakukan proses *sorting* nilai jarak yang diperoleh. 20% jarak terjauh dari total jumlah data akan dihapus. Nilai *round trip time* yang lolos dari *filtering* akan dijumlah dan dibagi dengan jumlah data yang tersisa untuk mendapatkan nilai rata-rata yang baru.

5.5 Implementasi proses perbandingan *RTT* dan deteksi

Proses perbandingan nilai *round trip time* dilakukan untuk menentukan *access point* yang dianggap sebagai *rogue access point*. *Access point* yang mendapatkan nilai *round trip time* lebih besar akan dideteksi sebagai *rogue*

access point. Implementasi proses melakukan perbandingan kedalam kode program diperlihatkan pada **gambar 5.6**.

```
if rata_rata > rata_rata2:
    print("access point 1 adalah RAP")
else:
    print("access point 2 adalah RAP")
```

Gambar 5.6 Implementasi perbandingan nilai *round trip time*

5.6 Implementasi proses penyimpanan data nilai RTT

Nilai rata-rata *round trip time* yang didapatkan dari kedua *access point* yang salah satunya dicurigai sebagai *rogue access point*, akan disimpan dalam sebuah file teks. Nilai rata-rata *round trip time* yang disimpan yakni nilai rata-rata *round trip time* awal sebelum dilakukan *filtering* dan rata-rata akhir setelah dilakukan *filtering*. Penyimpanan ini dilakukan untuk tujuan analisa dan perhitungan akurasi. Implementasi proses penyimpanan data nilai *round trip time* pada program ini diperlihatkan pada **gambar 5.7**.

```
def saveFile():
    print("\n\nSaving file...")
    fileSave = open("save6.txt", 'a')
    index = str(indexLoop+1)
    line = index+"\nRata-rata wifi asli = "+str(rata_rata)
    fileSave.write(line)
    line = "\nRata-rata RAP = "+str(rata_rata2)
    fileSave.write(line)
    line = "\nAvg 1 = "+str(first_avg)
    fileSave.write(line)
    line = "\nAvg 2 = "+str(first_avg2)+"\n\n"
    fileSave.write(line)
    fileSave.close()
```

Gambar 5.7 Implementasi proses penyimpanan nilai RTT

BAB 6 PENGUJIAN DAN ANALISIS

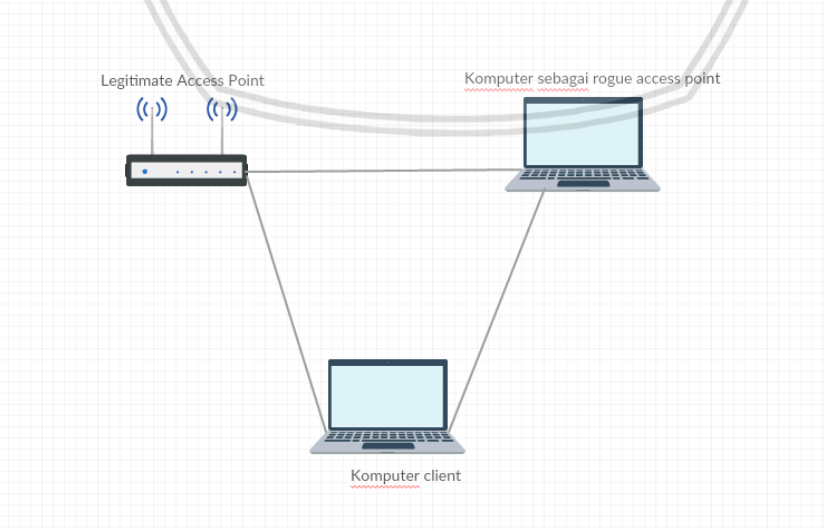
6.1 Pengujian

Pengujian dilakukan untuk mengetahui seberapa efektif metode perhitungan *round trip time* untuk mendeteksi adanya kemungkinan *rogue access point*. Program yang telah dibuat akan dijalankan untuk mendapatkan nilai dari *round trip time DNS* tiap *acces point* yang dicurigai sebagai *rogue access point*. Pengujian sistem akan dilakukan dengan terlebih dahulu mengimplementasikan lingkungan pengujian. Lingkungan pengujian sistem meliputi pembuatan *access point* dan instalasi *tool network shaper* untuk skenario pengujian ketika jaringan lambat.

Pengujian dilakukan dengan menghitung rata-rata nilai *round trip time* pada setiap *acces point* yang dicurigai sebagai *rogue access point*. Kemudian akan dibandingkan hasilnya. *Rogue access point* dengan tipe serangan *man-in-the-middle* yang memanfaatkan jaringan *acces point* asli kemudian melakukan imitasi dan menyebarkan jaringan palsu dengan *wireless card* eksternal secara teori akan menimbulkan nilai *round trip time* yang lebih besar yang diakibatkan oleh tambahan jumlah *hop*. Sehingga *acces point* dengan nilai *round trip time* lebih besar akan dideteksi sebagai *rogue access point*.

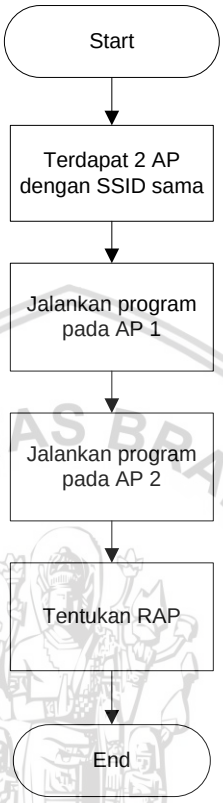
6.1.1 Perancangan lingkungan pengujian

Perancangan lingkungan pengujian digambarkan dengan topologi pada **gambar 6.1**. Pada **gambar 6.1** terlihat bahwa *rogue access point* diimplementasikan bahwa sebuah komputer yang terhubung pada *legitimate access point* dan memberikan koneksi internet pada pengguna lain dengan bertindak sebagai *rogue access point* yang seolah-olah menjadi *legitimate access point*.



Gambar 6.1 Topologi lingkungan pengujian

Untuk melakukan pengujian sebuah komputer yang telah diimplementasikan sistem pendeteksi *rogue access point* akan melakukan sambungan ke kedua *access point* yang identik kemudian menjalankan program sampai didapat hasil deteksi dan menentukan *access point access point* yang dianggap sebagai *rogue access point*. Alur kerja pengujian dijelaskan pada *flowchart* pada **gambar 6.2**.



Gambar 6.2 Alur kerja pengujian sistem

Dalam melakukan pengujian akan dibuat beberapa skenario untuk menghitung efektifitas dari program yang telah dibuat. Skenario pengujian yang akan dilakukan yaitu :

1. Pengujian pertama pada jaringan A yang diimplementasikan pada *access point* indiehome.
2. Pengujian kedua pada jaringan B dengan menggunakan sebuah komputer yang dikonfigurasikan sebagai *access point*.
3. Pengujian ketiga dengan mencatat nilai rata-rata *round trip time* sebelum dan sesudah proses *filtering* untuk melihat efektifitas dari proses *filtering*.
4. Pengujian keempat dengan menambahkan *delay* menggunakan Netem pada saat pendeteksian untuk melakukan simulasi jaringan yang tidak stabil.

6.1.2 Implementasi lingkungan pengujian

Implementasi lingkungan pengujian meliputi bagaimana *rogue access point* dibuat dan cara melakukan konfigurasi dan *routing* koneksi internet dari *rogue access point* ke pengguna atau korban.

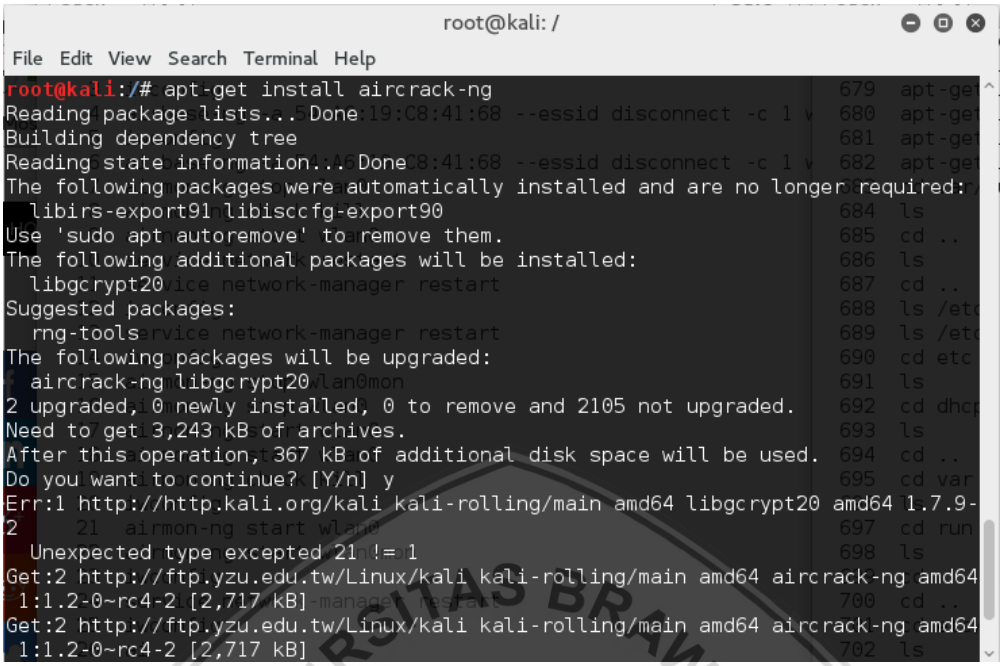
6.1.2.1 Implementasi *rogue access point*

Rogue access pont dibuat dengan menggunakan *tools* aircrack-ng yang dijalankan di sistem operasi kali linux. Aircrack-ng digunakan untuk membuat *rogue access point* yang identik dengan *legitimate access point*. *Tools* ini merupakan *tools* yang paling sering digunakan untuk membuat *rogue access point* karena mempunyai fitur yang cukup lengkap dan relatif mudah dalam melakukan implementasinya. Untuk mengimplementasikan serangan *rogue access point* akan digunakan sebuah laptop dengan 2 *wireless card*. Satu *wireless card* internal dari laptop dan satu *wireless card* tambahan menggunakan *usb external wireless card*. Salah satu *wireless card* akan digunakan untuk memberikan koneksi pada *client* sebagai *rogue access point,* dengan menggunakan *tools* aircrack-ng. *Rogue access point* akan menyebar sinyal dengan menyediakan koneksi internet dan menunggu korban yang tidak waspada untuk menyambung ke *rogue access point* yang telah dibuat. Sementara satu *wireless card* lainnya akan bertugas sebagai penghubung *rogue access point* ke koneksi internet yang disediakan *oleh legitimate access point*. Ilustrasi implementasi *rogue access point* diperlihatkan pada **gambar 6.3**.

Gambar 6.3 Ilustrasi implementasi *rogue access point*

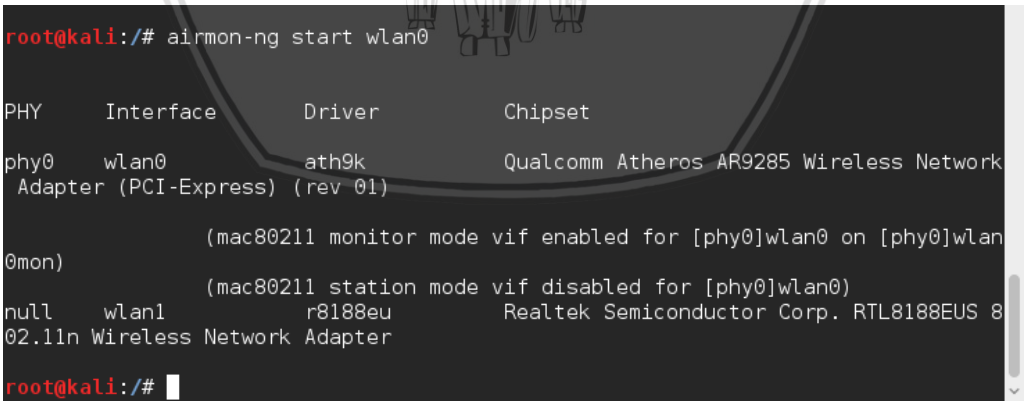
Aircrack-ng merupakan *tools* keamanan jaringan yang dapat digunakan untuk *monitoring* jaringan dan melakukan serangan seperti membuat *access point* palsu atau *rogue access point*. Fitur yang ada dalam *tools* aircrack-ng meliputi *packet sniffing*, mengatur *wireless card* sebagai *monitor mode*, membuat *access point* palsu dengan melakukan imitasi dari *access point* yang sebenarnya dan beberapa fitur lain yang terkait masalah *penetration testing* dan keamanan jaringan. Untuk menjalankan *tools* aircrack-ng pertama dilakukan penginstalan *tools* aircrack-ng dengan mengeksekusi perintah berikut pada terminal :

Apt-get install aircrack-ng



Gambar 6.4 Instalasi aircrack-ng

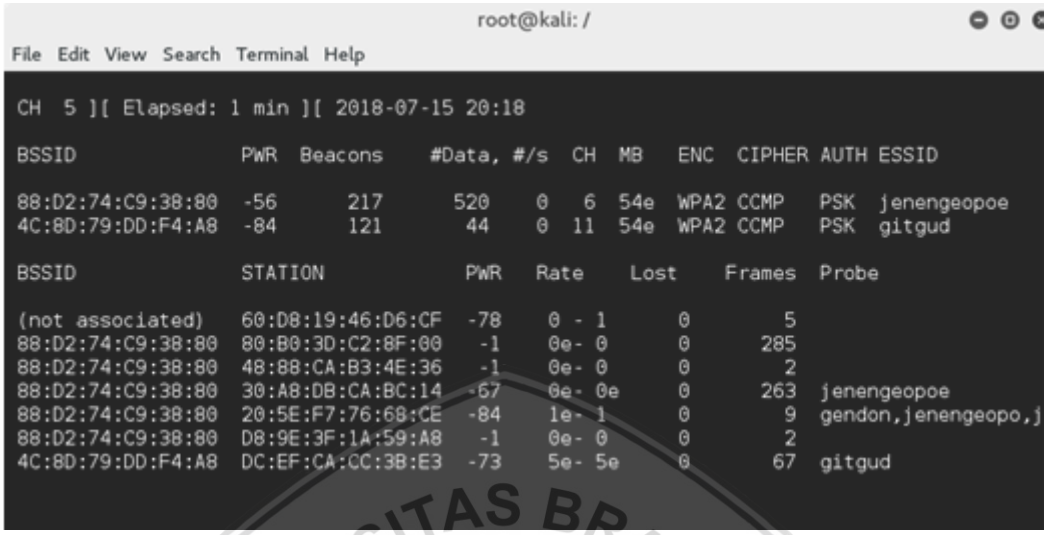
Pada **gambar 6.4** diperlihatkan proses instalasi *aircrack-ng.* Setelah melakukan penginstalan aircrack-ng selanjutnya mengeksekusi perintah *airmon-ng start wlan0.* Perintah airmon-ng merupakan perintah untuk membuat *wireless card* sebagai *monitor mode*. Wlan0 merupakan *interface wireless card* yang akan difungsikan menjadi *monitor mode* untuk kemudian melakukan *monitoring* trafik pada jaringan disekitar *station*. Pada **gambar 6.5** terlihat bahwa *interface* wlan0 telah berubah menjadi *monitor mode*.



Gambar 6.5 Eksekusi airmon-ng pada interface wlan0

Kemudian akan dilakukan *sniffing packet* untuk melihat trafik pada jaringan dengan melakukan eksekusi perintah *airodump-ng wlan0mon*. Dengan airodump-ng *attacker* bisa mengawasi trafik jaringan di sekitar. Paket yang dapat terlihat dari eksekusi perintah airodump-ng mulai dari BSSID (*mac address* dari *access point* yang ada disekitar jaringan), ESSID (nama *access point*), tipe enkripsi dan authentifikasi dari setiap *access point*, hingga *mac address* dari *station* yang

terkoneksi pada *access point* yang ada pada cakupan jaringan tersebut. Hasil paket yang dapat ditangkap oleh airodump-ng lebih lengkap dapat dilihat pada **gambar 6.6**.



Gambar 6.6 Capture dari perintah airodump-ng

Dari data yang didapat dengan eksekusi perintah airodump-ng akan terlihat BSSID dan ESSID yang akan digunakan untuk membuat imitasi dari access point yang asli untuk dibuat tiruannya. Dengan membuat *rogue access point* yang mengimitasi dari *legitimate access point*, tentu saja akan mengakibatkan *client* yang akan melakukan sambungan kemungkinan akan salah tersambung ke *rogue access point* yang telah dibuat. Akibatnya koneksi *client* bisa disadap dan dicurinya data-data yang penting.

Untuk membuat imitasi dari *legitimate access point* dilakukan dengan menggunakan perintah airbase-ng. *Access point* yang dibuat akan disesuaikan dari data yang ada dari hasil ekseskusi airodump-ng, dijalankan dengan eksekusi perintah berikut :

Airbase-ng –e jenengeopoe –c 6 wlan0mon

jenengeopoe digunakan sebagai contoh untuk membuat nama *access point* yang mirip ataupun bisa dibuat dengan nama yang sama dengan harapan *client* tidak dapat membedakan dan salah dalam melakukan sambungan ke *rogue access point* yang telah dibuat. Untuk membuat *rogue access point* mempunya *mac address* yang sama, digunakan *tool macchanger*. *Tool* ini dapat mengubah *mac address* dari *interface* wlan0 yang akan digunakan sebagai *rogue access point*. Untuk menggunakannya dieksekusi perintah :

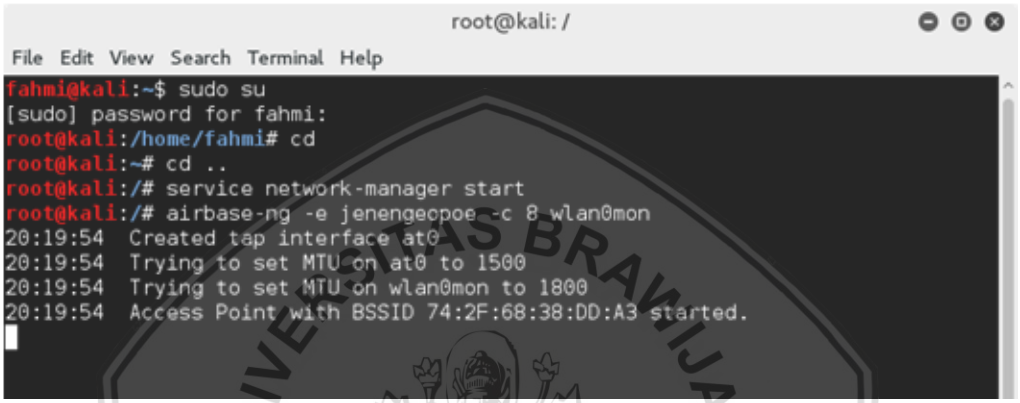
Ifconfig wlan0mon down

macchanger –m xx:xx:xx:xx:xx wlan0mon

ifconfig wlan0mon up

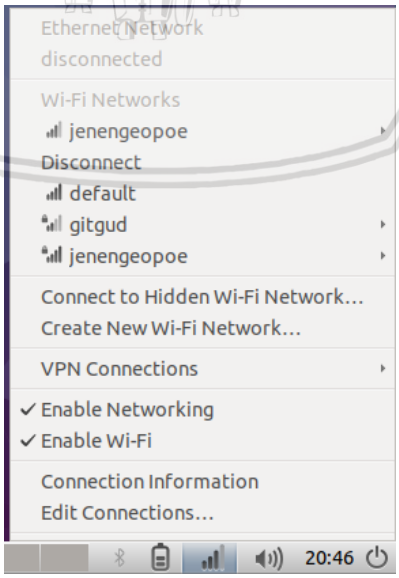
xx:xx:xx:xx:xx disesuaikan dengan *mac address* yang didapat dari *legitimate access point*.

Pada **gambar 6.7** diperlihatkan bahwa *access point* telah dibuat dan terlihat bahwa ada salah satu *station* yang berusaha menyambung pada *access point* yang telah dibuat. Pada sisi *client,* ketika mencari sambungan *wifi* yang *available* akan terlihat *rogue access point* dengan nama yang mirip atau identik dengan *legitimate access point*. Hal ini berakibat *client* bisa saja menyambung kepada *rogue access point* yang telah dibuat. Hasil *wifi available* yang terlihat pada komputer *client* diperlihatkan pada **gambar 6.8**. Terlihat pada **gambar 6.8** terdapat dua *access point* dengan nama "*jenengeopoe*". Bagi pengguna yang awam akan keamanan jaringa bisa jadi akan salah dalam melakukan sambungan sehingga membahayakan keamanan data pengguna.



Gambar 6.7 Membuat *rogue access point*

Tetapi pada tahap ini komputer *client* yang terhubung pada *rogue access point* yang telah dibuat masih belum bisa mendapat koneksi internet. Agar *client* yang tersambung bisa mendapatkan koneksi internet perlu dilakukan *routing* dan konfigurasi DHCP *server*. *Routing* dan konfigurasi DHCP *server* agar komputer *client* mendapatkan sambungan internet akan dijelaskan pada subbab **6.1.2.2**.

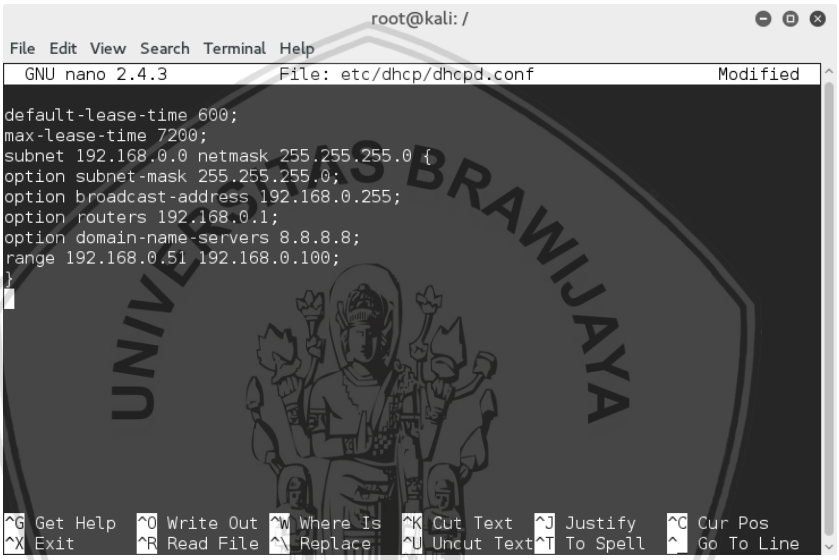


Gambar 6.8 *Wifi Available* dari sisi *client*

6.1.2.2 *Routing dan setting koneksi internet client*

Untuk memberikan koneksi internet pada *client* yang sudah terhubung pada *rogue access point* yang telah dibuat, dibutuhkan konfigurasi *routing* dan konfigurasi *dhcp server*. *Dhcp server* digunakan untuk memberikan *IP address* pada *client* yang nantinya akan terhubung pada *rogue access point*. Tanpa mendapatkan *IP address* tentu saja *client* tidak akan bisa tersambung ke internet.

Langkah pertama dalam menyediakan koneksi internet untuk *client* yang terhubung pada *rogue access point* yang telah dibuat yakni mengubah konfigurasi dhcpd.conf. File ini berada pada *path /etc/dhcp/dhcpd.conf*. Dengan menggunakan perintah *nano /etc/dhcp/dhcpd.conf*, pada file dhcpd.conf ditambahkan baris kode seperti yang terdapat pada **gambar 6.9.**

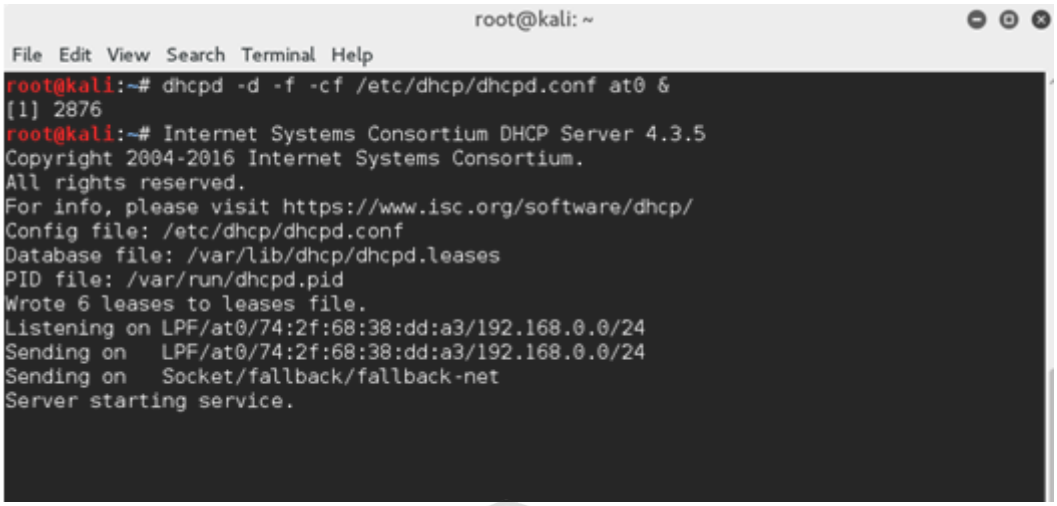


Gambar 6.9 konfigurasi dhcpd.conf

Pada baris terakhir file dhcpd.conf yakni baris kode *range* 192.168.0.51 192.168.0.100 merupakan batasan *ip address* yang bisa diperoleh *client* ketika nantinya akan terhubung pada *rogue access point* yang dibuat. Setelah melakukan konfigurasi dhcp.conf, *dhcp server* dijalankan dengan melakukan eksekusi perintah :

Dhcpd –d –f –cf /etc/dhcp/dhcpd.conf at0 &

Pada **gambar 6.10** diperlihatkan ketika dhcp *server* yang dibuat telah dapat berjalan.

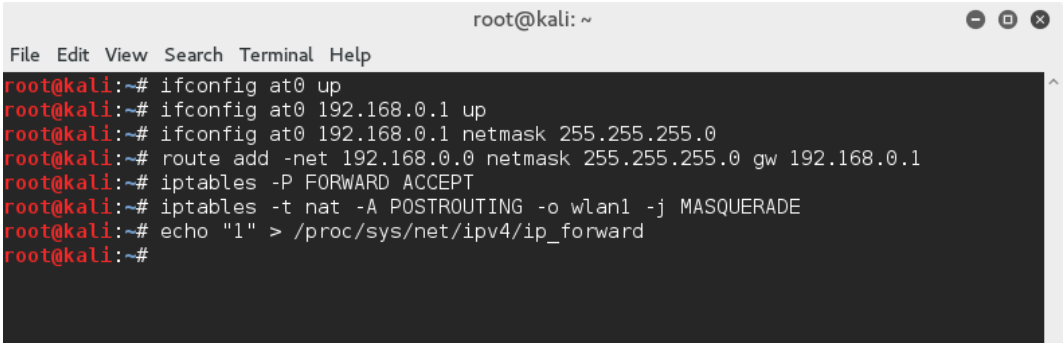


Gambar 6.10 DHCP *server* telah berjalan

Setelah konfigurasi dhcpd.conf dan menjalankan *dhcp server*, selanjutnya akan dieksekusi perintah sebagai berikut berturut-turut :

1. *Ifconfig at0 up*
2. *Ifconfig at0 192.168.0.1 up*
3. *Ifconfig at0 192.168.0.1 netmask 255.255.255.0*
4. *Route add –net 192.168.0.0 netmask 255.255.255.0 gw 192.168.0.1*
5. *Iptables –P FORWARD ACCEPT*
6. *Iptables –t nat –A POSTROUTING –o wlan1 –j MASQUERADE*
7. *Echo "1" > /proc/sys/net/ipv4/ip_forward*

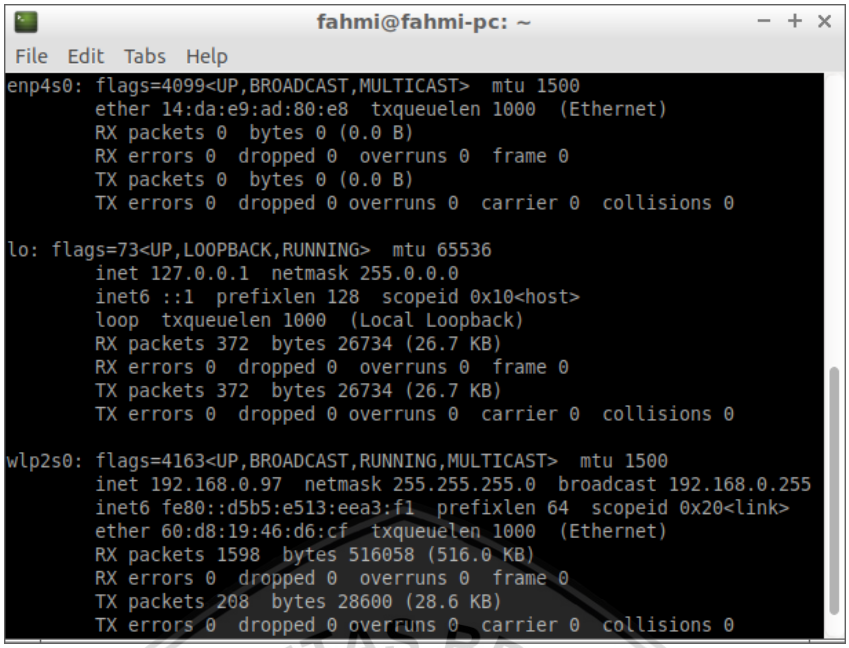
Pada **gambar 6.11** diperlihatkan proses *routing* mulai dari konfigurasi *interface* at0, konfigurasi *firewall* dan *enabling ip forwarding*. Perintah pada baris pertama dilakukan untuk inisialisasi *interface at0. Ip address* untuk *interface at0* dikonfigurasikan pada alamat 192.168.0.1. Perintah pada baris keempat digunakan untuk melakukan konfigurasi rute semua paket akan dilewatkan pada alamat 192.168.0.0 dengan *gateway* 192.168.0.1. Pada baris kelima digunakan untuk melakukan *disable* sementara atau *reset rules firewall* yang ada sebelumnya. Baris keenam merupakan konfigurasi paket yang ditujukan pada *rogue access point* akan dilewatkan pada *interface wlan1* yang mempunyai akses ke *legitimate access point* sehingga pengguna dapat tersambung ke internet. Baris ketujuh digunakan untuk mengaktifkan *ip forwarding.* Pada **gambar 6.12** terlihat bahwa *ip address* dan *subnetmask* dari at0 sudah berubah sesuai dengan yang telah dikonfigurasikan. At0 merupakan *interface* dari *rogue access point* yang dibuat oleh perintah airbase-ng. Setelah konfigurasi selesai *client* sudah dapat terhubung dengan internet dan dapat menggunakan internet dengan normal.



Gambar 6.11 *Routing*

Gambar 6.12 Detail ifconfig

Pada **gambar 6.13** terlihat bahwa *client* sudah tersambung dengan *rogue access point* yang dibuat dan mendapatkan koneksi internet. *IP address* yang didapatkan oleh *client* adalah 192.168.0.97. *Ip address* ini masuk dalam *range IP address* yang didefinisikan pada konfigurasi *file* dhcpd.conf yakni *range IP address client* dibatasi mulai dari 192.168.0.51 sampai 192.168.0.100.



Gambar 6.13 Detail koneksi internet dari sisi *client*

6.1.3 Pengujian Program Perhitungan *Round Trip Time*

Pada pengujian program akan menjalankan program yang telah dibuat. Dengan pengaturan *DNS* yang digunakan adalah *DNS non-recursive*. Pengujian akan menjalankan program pada kedua *access point* dengan SSID dan *MAC address* yang sama. Kemudian perbandingan hasil keduanya akan menentukan *access point* mana yang akan dideteksi sebagai *rogue access point* karena perbedaan jumlah *hop* dari *rogue access point* akan menimbulkan perbedaan waktu yang tidak bisa dihindari.

6.1.3.1 *Pengujian program*

Setelah dideteksi terdapat akses point dengan SSID dan *MAC address* yang sama program akan mulai dijalankan. Program akan dijalankan dan menghubungkan *computer client* ke salah satu *access point* yang dicurigai sebagai *rogue access point*. Program melakukan perhitungan *round trip time* dari masing-masing *domain* atau *website* yang akan diuji lalu melakukan perhitungan rata-rata dan menjalankan proses *filtering*. Rata-rata akhir akan disimpan. Setelah selesai mendapatkan rata-rata akhir program akan melakukan sambungan ke *access point* kedua yang mempunyai SSID sama dan dicurigai sebagai *rogue access point*. Program selanjutnya melakukan perhitungan yang sama sampai didapatkan rata-rata akhir. Rata-rata akhir dari *access point* satu dan *access point* dua akan dibandingkan. Sesuai teori seharusnya akan ada *access point* dengan nilai *round trip time* yang lebih besar dikarenakan perbedaan jumlah *hop* yang ditimbulkan.

Untuk menghitung berapa persentase akurasi program dalam melakukan deteksi *rogue access point*, program akan dijalankan 100 kali. Dalam 100 kali program berjalan akan didapatkan berapa kali program dapat mengidentifikasi

rogue access point dengan benar dan berapa kali program salah dalam mengidentifikasi *rogue access point*. Kesalahan dalam pendeteksian *rogue access point* bisa terjadi dikarenakan kondisi jaringan yang digunakan tidak stabil. Karena pendeteksian *rogue access point* dalam penelitian ini menggunakan metode perhitungan waktu otomatis akan sangat sensitif terhadap *delay*. Proses *filtering* digunakan untuk meminimalisasi kesalahan pendeteksian. Pengujian program dijalankan 2 kali pada jaringan yang berbeda. Program akan dijalankan sekali terlebih dahulu dengan mengabaikan nilai RTT. Hal ini dilakukan agar *DNS* tersimpan dalam *cache local DNS server* sehingga waktu dalam melakukan deteksi lebih stabil. Penggunaan *cache local DNS server* bertujuan untuk mengurangi waktu yang dibutuhkan oleh *DNS lookup* jika harus mengontak *root server.*

6.1.3.2 *Pengujian Proses Filtering*

Filtering digunakan untuk mengurangi dampak yang ditimbulkan oleh kondisi jaringan yang tidak stabil. Karena penelitian ini menggunakan jaringan *real*, tentu terkadang kondisi jaringan bisa jadi tiba-tiba lambat yang akan mempengaruhi nilai *round trip time* yang bisa jadi tiba-tiba menjadi besar. Hal ini akan berpengaruh terhadap akurasi pendeteksian *rogue access point*.

Untuk menguji seberapa efektif *filtering* dapat meningkatkan akurasi pendeteksian *rogue access point*, program akan dijalankan dan menyimpan rata-rata nilai *round trip time* awal sebelum proses *filtering* dilakukan dan rata-rata akhir ketika proses *filtering* telah dilakukan. Dari kedua data yang dihasilkan akan dapat dianalisis berapa persentase akurasi pendeteksian dengan *filtering* dan berapa persentase akurasi pendeteksian tanpa *filtering*. Selain itu akan dilakukan pengujian dengan simulasi kondisi jaringan yang tiba-tiba lambat. Untuk itu akan digunakan *tool network shaper* dengan menggunakan *Netem* untuk menambah *delay* atau melakukan simulasi koneksi lambat.

6.1.4 Hasil pengujian sistem deteksi *rogue access point*

Dalam pengujian sistem pendeteksian *rogue access point* yang telah dilakukan didapatkan hasil seperti yang diperlihatkan pada **tabel 5.1**. Pengujian dilakukan dengan menggunakan *round trip time* sebagai *parameter*. Paket yang dikirim untuk mendapatkan waktu *round trip time* yakni *DNS lookup*. Pengujian menggunakan 50 *domain* atau *website*. Jumlah *domain* yang digunakan sebanyak 50 karena dianggap lebih ideal. Terlalu banyak jumlah *domain* akan mempengaruhi waktu deteksi yang lebih lama. Sedangkan jika terlalu sedikit ketika koneksi internet tiba-tiba lambat dalam waktu cukup lama, proses *filtering* tidak mampu bekerja maksimal dalam menghapus nilai *round trip time* tinggi yang dihasilkan oleh hal tersebut. Daftar *website* didapatkan dari *history browser*. Seluruh *domain* dipastikan *valid* dan dapat diakses untuk memastikan bahwa *round trip time* yang didapatkan benar-benar *valid*.

Tabel 6.1 Hasil Pengujian Deteksi *Rogue Access Point*

No	Rata-rata wifi asli	Rata-rata RAP	Selisih
1	11,85365854	20,375	8,521341463
2	12,09756098	11,725	-0,372560976
3	8,256097561	14,2	5,943902439
4	9,487804878	11,375	1,887195122
5	9,914634146	14,45	4,535365854
6	11,74390244	12,375	0,631097561
7	8,756097561	13,95	5,193902439
8	8,5	21,6	13,1
9	13,29268293	26,35	13,05731707
10	12,7804878	11,95	-0,830487805
11	7,719512195	24,2	16,4804878
12	20,26829268	23,675	3,406707317
13	17,7804878	16,275	-1,505487805
14	17,06097561	20,575	3,51402439
15	10,40243902	16,775	6,372560976
16	15,95121951	18,575	2,623780488
17	12,06097561	17,85	5,78902439
18	13,91463415	13,11	-0,804634146
19	5,025	63,72	58,695
20	3,025	13,47	10,445
21	13,1097561	13,375	0,265243903
22	10,32926829	13,14	2,810731707
23	14,06097561	23,5	9,43902439
24	21,45121951	27,32	5,868780488
25	10	18	8
26	10,46341463	19,68	9,216585366
27	8,280487805	12,57	4,289512195
28	15,40243902	18,875	3,472560976
29	24,29268293	21,2	-3,092682927
30	14,37804878	51,2	36,82195122
31	12,1097561	14,95	2,840243903
32	10,8902439	34,775	23,8847561
33	12,63414634	22,775	10,14085366
34	12,26829268	17,775	5,506707317
35	15,7195122	14,65	-1,069512195
36	11,5	19,9	8,4
37	10,6097561	25,55	14,9402439
38	20,98780488	30,85	9,862195122
39	28,95121951	30	1,048780488
40	15,12195122	13,625	-1,49695122

Tabel 6.1 Hasil Pengujian Deteksi *Rogue Access Point* (Lanjutan)

No	Rata-rata wifi asli	Rata-rata RAP	Selisih
41	15,75609756	23,25	7,493902439
42	19,30487805	19,825	0,520121951
43	11,59756098	18,775	7,177439024
44	10,40243902	13,55	3,147560976
45	12,24390244	12,15	-0,093902439
46	9,268292683	17,6	8,331707317
47	19,15853659	34,07	14,91146341
48	3,475	13,97	10,495
49	10,07317073	22,4	12,32682927
50	9,109756098	15,85	6,740243902
51	9,951219512	13,85	3,898780488
52	9,317073171	19,2	9,882926829
53	13,31707317	22,87	9,552926829
54	16,32926829	27,55	11,22073171
55	21,17073171	22,32	1,149268293
56	12,52439024	17,77	5,245609756
57	15,18292683	17,67	2,487073171
58	10,20731707	14,82	4,612682927
59	11,04878049	12,875	1,826219512
60	9,109756098	15,85	6,740243902
61	9,951219512	13,85	3,898780488
62	9,317073171	19,2	9,882926829
63	11,85365854	20,375	8,521341463
64	12,81707317	27,8	14,98292683
65	15,7195122	23,875	8,155487805
66	11,76829268	13,15	1,381707317
67	8,402439024	11,425	3,022560976
68	8,280487805	16,525	8,244512195
69	11,92682927	13,4	1,473170732
70	8,780487805	22,2	13,4195122
71	16,58536585	27,1	10,51463415
72	6,987804878	11,851	4,863195122
73	9,707317073	11,6	1,892682927
74	7,634146341	15	7,365853659
75	10,41463415	16,25	5,835365854
76	15,30487805	14,475	0,829878049
77	9,634146341	20,2	10,56585366
78	20,42682927	26,2	5,773170732
79	11,59756098	14,925	3,327439024
80	14,3902439	16,9	2,509756098

Tabel 6.1 Hasil Pengujian Deteksi *Rogue Access Point* (Lanjutan)

No	Rata-rata wifi asli	Rata-rata RAP	Selisih
81	12,57317073	15,525	2,951829268
82	11,97560976	14,75	2,774390244
83	18,65853659	17,075	1,583536585
84	9,231707317	14,275	5,043292683
85	9,292682927	14,625	5,332317073
86	10,75609756	14,2	3,443902439
87	11,48780488	13,5	2,012195122
88	9,012195122	15,375	6,362804878
89	8,817073171	18,275	9,457926829
90	10,47560976	12,775	2,299390244
91	10,34146341	21	10,65853659
92	7,524390244	12,425	4,900609756
93	8,573170732	13,225	4,651829268
94	9,756097561	15,7	5,943902439
95	7,43902439	12,6	5,16097561
96	8,719512195	13	4,280487805
97	9,792682927	14,2	4,407317073
98	9,219512195	14,775	5,555487805
99	9,548780488	12,85	3,301219512
100	9,243902439	13,52	4,276097561
	rata-rata wifi asli	rata-rata RAP	
	12,04939634	18,56471	6,515313659

Pengujian kedua dilakukan pada jaringan yang berbeda yakni dengan menggunakan sebuah komputer yang digunakan sebagai *access point*. Hal ini untuk memastikan bahwa program mampu berjalan dengan baik di beberapa jaringan berbeda. Pada pengujian di jaringan yang kedua didapatkan hasil seperti pada **tabel 6.2**.

Tabel 6.2 Hasil pengujian pada jaringan kedua

No	Rata-rata wifi asli	Rata-rata RAP	selisih
1	12,75	18,875	6,125
2	10,95	17,525	6,575
3	9,975	23,8	13,825
4	8,5	23,15	14,65
5	10,45	17,325	6,875
6	8,175	13,5	5,325
7	3,5	10	6,5
8	11,05	18,575	7,525

Tabel 6.2 hasil pengujian pada jaringan kedua (lanjutan)

No	Rata-rata wifi asli	Rata-rata RAP	selisih
9	3,175	22,275	19,1
10	13,125	19,95	6,825
11	10,325	18,15	7,825
12	9,5	17,275	7,775
13	5	17,175	12,175
14	9,125	19,025	9,9
15	12,2	37,4	25,2
16	9,275	20,25	10,975
17	49,05	60,225	11,175
18	11,85	14,8	2,95
19	3,725	9,875	6,15
20	7,426829268	12,1	4,673170732
21	8,365853659	12,4	4,034146341
22	8,036585366	9,925	1,888414634
23	3,025	7,125	4,1
24	7,585365854	11,5	3,914634146
25	10,53658537	13,25	2,713414634
26	10,26829268	10,175	-0,093292683
27	7,109756098	10,375	3,265243902
28	4,35	10,425	6,075
29	7,585365854	14,25	6,664634146
30	3,325	12,875	9,55
31	12,2195122	17,2	4,980487805
32	8,963414634	17,9	8,936585366
33	16,025	15	-1,025
34	9,93902439	14,05	4,11097561
35	9,304878049	11,05	1,745121951
36	8,170731707	14,3	6,129268293
37	3,55	13,275	9,725
38	9,097560976	11,05	1,952439024
39	7,048780488	10,375	3,326219512
40	3,425	16,575	13,15
41	3,95	12,925	8,975
42	8,743902439	11,575	2,831097561
43	8,853658537	15,1	6,246341463
44	3,275	11,1	7,825
45	7,06097561	11,775	4,71402439
46	8,914634146	14,75	5,835365854
47	8,402439024	12,85	4,447560976
48	8,951219512	10,525	1,573780488

Tabel 6.2 hasil pengujian pada jaringan kedua (lanjutan)

No	Rata-rata wifi asli	Rata-rata RAP	selisih
49	7,243902439	12,7	5,456097561
50	13,68292683	12,475	-1,207926829
51	8,207317073	11,4	3,192682927
52	8,658536585	15,075	6,416463415
53	7,926829268	10,575	2,648170732
54	3,225	17,075	13,85
55	10,58536585	17,225	6,639634146
56	12,36585366	17,85	5,484146342
57	10,91463415	12,525	1,610365854
58	8,170731707	19,05	10,87926829
59	10,48780488	16,075	5,587195122
60	10,73170732	34,85	24,11829268
61	18,86585366	18,275	-0,590853659
62	10,74390244	13,4	2,656097561
63	9,085365854	15,5	6,414634146
64	8,207317073	20,375	12,16768293
65	12,40243902	14,525	2,122560976
66	9,182926829	17,15	7,967073171
67	9,292682927	16,975	7,682317073
68	11,62195122	16,25	4,628048781
69	10,73170732	14	3,268292683
70	9,353658537	15,825	6,471341463
71	10,18292683	18,375	8,192073171
72	14,80487805	13,625	-1,179878049
73	8,890243902	16,6	7,709756098
74	12,76829268	13,825	1,056707317
75	10,01219512	11,95	1,937804878
76	8,073170732	14,5	6,426829268
77	9,670731707	17,225	7,554268293
78	10,90243902	22,3	11,39756098
79	3,025	19,925	16,9
80	9,109756098	12,575	3,465243902
81	10,40243902	12,9	2,497560976
82	8,865853659	14,2	5,334146341
83	9,914634146	13,15	3,235365854
84	10,86585366	19,525	8,659146342
85	10,2804878	12,5	2,219512195
86	8,768292683	21,575	12,80670732
87	14,1097561	15,5	1,390243902
88	9,170731707	13,95	4,779268293

Tabel 6.2 hasil pengujian pada jaringan kedua (lanjutan)

No	Rata-rata wifi asli	Rata-rata RAP	selisih
89	11,48780488	14,85	3,362195122
90	9,524390244	12,75	3,225609756
91	8,658536585	14,4	5,741463415
92	7,951219512	16,175	8,223780488
93	9,695121951	15,425	5,729878049
94	9,097560976	18,8	9,702439024
95	10,7804878	13,85	3,069512195
96	10,14634146	22,975	12,82865854
97	13,69512195	14,55	0,854878049
98	11,01219512	14,825	3,812804878
99	12,37804878	13,75	1,37195122
100	3,175	15,175	12
	Rata-Rata Wifi Asli	Rata-rata RAP	Rata-rata selisih
	9,543182927	15,958	6,414817073

6.1.5 Hasil pengujian proses *filtering*

Proses *filtering* digunakan untuk menyaring nilai *round trip time* yang anomali atau berbeda. Nilai yang anomali ini bisa didapatkan dari kondisi jaringan yang tidak stabil. Ketika jaringan tiba-tiba sibuk atau mengalami gangguan dalam jeda waktu tertentu, akan mempengaruhi hasil pengujian. Nilai *round trip time* yang meninggi pada waktu tertentu akan membuat terjadinya kemungkinan program salah dalam melakukan deteksi. Proses *filtering* digunakan untuk meminimalisir terjadinya hal tersebut dan juga digunakan untuk meningkatkan akurasi pendeteksian *rogue access point*. Dalam **tabel 6.3** ditunjukkan hasil pengujian seberapa efektif proses *filtering* dapat meningkatkan akurasi pendeteksian *rogue access point*.

Tabel 6.3 Hasil pengujian fungsi *filtering*

No	Rata-rata RTT Wifi asli	Rata-rata RTT RAP	RTT Sebelum *Filtering* wifi asli	RTT Sebelum *Filtering RAP*	Selisih RTT	Selisih RTT sebelum *filtering*
1	3	12,125	3,18	31,18	9,125	28
2	3,1	10,625	3,72	16,2	7,525	12,48
3	3,525	12,6	3,94	116,64	9,075	112,7
4	3,65	10,125	4,06	11,12	6,475	7,06
5	3,2	10,15	3,34	10,38	6,95	7,04
6	3,125	11,775	3,02	34,28	8,65	31,26
7	3,45	12,825	4,1	27,2	9,375	23,1

Tabel 6.3 Hasil pengujian fungsi *filtering*

No	Rata-rata RTT Wifi asli	Rata-rata RTT RAP	RTT Sebelum *Filtering* wifi asli	RTT Sebelum *Filtering RAP*	Selisih RTT	Selisih RTT sebelum *filtering*
8	3,575	14,7	4,3	92,4	11,125	88,1
9	3,25	13,55	4,9	24,8	10,3	19,9
10	3,825	12,425	4,08	29,1	8,6	25,02
11	3,15	13,025	3,38	52,54	9,875	49,16
12	4,325	13,075	7,46	29,54	8,75	22,08
13	3,175	9,85	3,2	10,72	6,675	7,52
14	3,325	11,8	3,82	17,02	8,475	13,2
15	3,175	12,3	3,74	29,38	9,125	25,64
16	3,25	12,775	5,04	48,48	9,525	43,44
17	4,05	10,325	6,6	11,14	6,275	4,54
18	3,3	11,475	4,82	31,06	8,175	26,24
19	4,1	11,725	7,76	32,56	7,625	24,8
20	3,075	11,475	3,32	14,96	8,4	11,64
21	3,375	11,975	4,48	89,04	8,6	84,56
22	3,35	14,85	5,7	84,24	11,5	78,54
23	3,675	10,85	5,66	14,84	7,175	9,18
24	3,625	22,85	5,3	88,74	19,225	83,44
25	3,425	14,9	5,46	85,6	11,475	80,14
26	4,95	16,95	6,2	99,3	12	93,1
27	3,3	10,5	5,06	12,84	7,2	7,78
28	3,725	16	4,76	78,86	12,275	74,1
29	3,2	12,825	3,82	32,52	9,625	28,7
30	2,975	11,45	3,14	27,56	8,475	24,42
31	2,9	11,125	2,96	13,56	8,225	10,6
32	3,275	21,5	4,9	82,66	18,225	77,76
33	3,6	10,325	6,38	14,34	6,725	7,96
34	3,225	10,4	4,5	28,66	7,175	24,16
35	3,375	11,075	4,4	13,94	7,7	9,54
36	3,075	14,35	3,3	70,16	11,275	66,86
37	3,175	13,075	3,18	32,7	9,9	29,52
38	3,05	10,75	3,24	19,56	7,7	16,32
39	3.175	13.425	3.58	46.28	10.25	42.7
40	3.425	13.875	3.64	50.18	10.45	46.54
41	3.125	10.075	3.2	11.22	6.95	8.02
42	3.75	12.825	13.18	31.02	9.075	17.84
43	3.4	11.25	4.72	53.5	7.85	48.78
44	3.925	10.5	6.72	12.98	6.575	6.26
45	3.375	11.3	3.94	31.22	7.925	27.28

Tabel 6.3 hasil pengujian fungsi *filtering* (lanjutan)

No	Rata-rata RTT Wifi asli	Rata-rata RTT RAP	RTT Sebelum *Filtering* wifi asli	RTT Sebelum *Filtering RAP*	Selisih RTT	Selisih RTT sebelum *filtering*
46	7.35	17.725	25	41.52	10.375	16.52
47	9.8	22.175	14.4	110.28	12.375	95.88
48	8.575	15.625	14.42	105.12	7.05	90.7
49	3.775	18.4	4.46	115.9	14.625	111.44
50	8.7	21.025	15.2	88.4	12.325	73.2
51	8.525	16.675	11.5	87.42	8.15	75.92
52	8.3	14.65	16.22	84.1	6.35	67.88
53	6.05	13.45	10.16	80.76	7.4	70.6
54	9.025	16.075	14.06	93.86	7.05	79.8
55	7.975	13.375	13.02	81	5.4	67.98
56	8.6	21.15	15.6	46.88	12.55	31.28
57	9.925	20.275	14.68	126.86	10.35	112.18
58	10.4	17.25	16.98	96.92	6.85	79.94
59	8.225	20.1	12.48	134.52	11.875	122.04
60	8.525	20.45	12.52	111.54	11.925	99.02
61	8.925	14.225	12.24	86.86	5.3	74.62
62	7.95	20.225	10.18	124.38	12.275	114.2
63	7.925	19.025	11.74	89	11.1	77.26
64	8.3	14.3	14.72	73.54	6	58.82
65	8.75	12.525	10.8	37	3.775	26.2
66	8.225	16.8	11.4	68.16	8.575	56.76
67	8.825	19.675	16.04	93.84	10.85	77.8
68	10.5	16.425	20.26	39.66	5.925	19.4
69	7.725	16.625	8.94	52.6	8.9	43.66
70	12.7	19.775	17.9	56.92	7.075	39.02
71	12.75	18.275	15.66	68.58	5.525	52.92
72	10.375	24.425	21.16	163.3	14.05	142.14
73	9.825	15.925	14.7	137.94	6.1	123.24
74	10.625	18.75	18.24	86.36	8.125	68.12
75	8.475	12.35	11.42	22.3	3.875	10.88
76	8.3	14.375	11.16	56.66	6.075	45.5
77	8,225	15,375	10,86	183,82	7,15	172,96
78	7,5	19,625	10,08	104,54	12,125	94,46
79	11,425	15,55	19,68	122,96	4,125	103,28
80	10	19,825	15,72	108,34	9,825	92,62
81	8,35	16,75	14,14	68,78	8,4	54,64
82	12,5	12,875	33,82	32,52	0,375	-1,3
83	10,975	13,6	15,4	53,82	2,625	38,42

Tabel 6.3 hasil pengujian fungsi *filtering* (lanjutan)

No	Rata-rata RTT Wifi asli	Rata-rata RTT RAP	RTT Sebelum *Filtering* wifi asli	RTT Sebelum *Filtering RAP*	Selisih RTT	Selisih RTT sebelum *filtering*
84	11,45	21,75	15,18	114,58	10,3	99,4
85	11,25	10,325	21,08	10,84	-0,925	-10,24
86	16,775	13,625	40,96	36,04	-3,15	-4,92
87	13,075	14,4	32,66	51,76	1,325	19,1
88	7,8	10,15	10,3	12,46	2,35	2,16
89	11,15	11,725	16,6	48,52	0,575	31,92
90	9,775	12,2	22,16	64,14	2,425	41,98
91	8,375	13,05	10,48	169,46	4,675	158,98
92	10,125	12,725	15,2	28,14	2,6	12,94
93	10,35	13,475	15,68	29,54	3,125	13,86
94	11,275	19,325	16,84	87,02	8,05	70,18
95	10,9	14,4	15,08	22,92	3,5	7,84
96	8,125	10,5	9,84	16,94	2,375	7,1
97	9,2	12,375	13,86	26,72	3,175	12,86
98	10,7	10,425	22,18	12,36	-0,275	-9,82
99	8,55	12,175	11,68	29,26	3,625	17,58
100	10,175	10,225	23	11,16	0,05	-11,84
	Rata-rata Asli	rata-rata RAP			Selisih	
	6,76025	14,46275			7,7025	

Pengujian *filtering* kedua dilakukan dengan menggunakan bantuan *Netem* untuk menambah *delay* ketika program melakukan perhitungan *round trip time* pada *legitimate access point*. Pengujian dilakukan dengan menjalankan program sebanyak 5 kali dengan penambahan delay 500 ms ketika program melakukan perhitungan *round trip time* pada *legitimate access point*. Hasil yang didapat diperlihatkan pada **tabel 6.4**.

Tabel 6.4 Pengujian *filtering* dengan *Netem*

No	Rata-rata RTT Wifi asli	Rata-rata RTT RAP	Selisih RTT	RTT Sebelum *Filtering* wifi asli	RTT Sebelum *Filtering RAP*	Selisih RTT sebelum *filtering*
1	3,95	12,27	8,32	95,44	47,72	-47,72
2	4,55	11	6,45	139,06	18,08	-120,98
3	3,5	10,6	7,1	81,32	10,68	-70,64
4	5,675	10,25	4,575	52,16	12,56	-39,6
5	5,5	9,925	4,425	50,34	11,14	-39,2

6.2 Analisis Hasil Pengujian

6.2.1 Analisis hasil pengujian sistem pendeteksian *rogue access point*

Dari pengujian yang telah dilakukan dapat terlihat pada **tabel 6.1**, nilai akurasi yang diperoleh mencapai 90%. Kesalahan pendeteksian disebabkan oleh kondisi jaringan yang tidak stabil sewaktu menjalankan program. Dari 100 kali percobaan terdapat 10 kali program salah dalam melakukan deteksi *rogue access point*. Rata-rata nilai *round trip time* yang didapatkan dari *legitimate access point* adalah 12,04939634 ms. Sementara untuk rata-rata *round trip time* dari *rogue access point* didapatkan angka 18,56471 ms. Jadi rata-rata selisih yang didapatkan dari *round trip time legitimate access point* dan *rogue access point* adalah 6,515313659 ms.

Dari **tabel 6.1** terlihat pada data baris ke 39 ketika nilai *round trip time legitimate access point* melonjak menjadi 28 ms, nilai *round trip time* pada *rogue access point* juga melonjak menjadi 30 ms. Hal ini disebabkan karena dalam suatu waktu ketika lalu lintas jaringan menjadi padat dan menyebabkan nilai *round trip time* meninggi tidak akan menjadi masalah dalam pendeteksian asalkan trafik yang tinggi tersebut terjadi dalam *range* waktu pengambilan nilai *round trip time* dari kedua *access point*. Hal ini bisa jadi bermasalah ketika trafik meninggi pada pengambilan nilai *round trip time* pada *legitimate access point* dan kemudian kembali normal pada pengambilan nilai *round trip time* pada *rogue access point*. Hal tersebut tentunya akan membuat nilai *round trip time* dari *legitimate access point* akan menjadi lebih besar daripada nilai *round trip time* dari *rogue access point*. 10 kali kesalahan pendeteksian terjadi dimungkinkan karena adanya hal tersebut.

Pada jaringan kedua dibuat dengan menggunakan sebuah laptop yang digunakan sebagai *access point* yang asli. Pada pengujian kedua ini didapatkan hasil akurasi mencapai 95%. Hasil pengujian ini diperlihatkan pada **tabel 6.2.** Pada pengujian jaringan kedua rata-rata yang diperoleh dari *access point* asli adalah 9,543182927 ms. Sementara pada *rogue access point* rata-rata *round trip time* yang diperoleh adalah 15,958 ms. Selisih waktu *round trip time* yang dihasilkan mencapai 6,414817073 ms. Kesalahan pendeteksian terjadi pada data ke 26, 33, 50, 61 dan 72. Pada pengujian kedua, selisih waktu *round trip time* yang dihasilkan ketika ada kesalahan pendeteksian tidak pernah lebih dari 1,5 ms.

Tabel 6.5 dan **6.6** merupakan tabel *confusion matrix* yang merepresentasikan hasil pengujian yang telah dilakukan pada jaringan pertama maupun kedua. *Confusion matrix* digunakan untuk mempermudah pembacaan hasil pengujian yang telah dilakukan. **Tabel 6.5** merupakan representasi dari data yang diperoleh pada **tabel 6.1**. Sementara **tabel 6.6** merupakan representasi data yang diperoleh dari hasil pengujian **tabel 6.2**.

Tabel 6.5 *confusion matrix* hasil pengujian pada jaringan pertama

N =100		Predicted	
		Rogue access point	*Legitimate access point*
Actual	*Rogue access point*	90	10
	Legitimate access point	10	90

Tabel 6.6 *confusion matrix* hasil pengujian pada jaringan kedua

N =100		Predicted	
		Rogue access point	*Legitimate access point*
Actual	*Rogue access point*	95	5
	Legitimate access point	5	95

Pada *confusion matrix* **tabel 6.5** diperlihatkan hasil yang diperoleh saat pengujian pada jaringan pertama. Program mendeteksi mampu *legitimate access point* sebagai *legitimate access point* atau *true positives* sebanyak 90 kali dari 100 data. Sementara *false positives* atau program melakukan pendeteksian *legitimate access point* sebagai *rogue access point* sebanyak 10 kali. Program mendapatkan nilai *true negatives* atau mendeteksi *rogue access point* sebagai *rogue access point* sebanyak 90 kali. Dan *false negative* atau mendeteksi *rogue access point* sebagai *legitimate access point* sebanyak 10 kali.

Pada pengujian jaringan yang kedua data hasil pengujian direpresentrasikan pada *confusion matrix* **tabel 6.6**. Dalam tabel ini terdapat peningkatan akurasi yang dihasilkan. Program Program mendeteksi mampu *legitimate access point* sebagai *legitimate access point* atau *true positives* sebanyak 95 kali dari 100 data. Sementara *false positives* atau program melakukan pendeteksian *legitimate access point* sebagai *rogue access point* sebanyak 5 kali. Program mendapatkan nilai *true negatives* atau mendeteksi *rogue access point* sebagai *rogue access point* sebanyak 95 kali. Dan *false negative* atau mendeteksi *rogue access point* sebagai *legitimate access point* sebanyak 5 kali.

6.2.2 Analisis hasil pengujian proses *filtering*

Proses *filtering* digunakan untuk menyaring nilai *round trip time* yang anomali. Dalam hal ini *filtering* digunakan untuk menghapus nilai *round trip time* yang tiba-tiba membesar yang dipengaruhi oleh kondisi jaringan yang sedang lambat. Pada pengujian proses *filtering* yang diperlihatkan pada **tabel 6.3**, terlihat bahwa nilai akurasi deteksi *rogue access point* sebelum dilakukannya *filtering* didapatkan angka 95%. Sementara nilai akurasi pendeteksian *rogue access point* setelah proses *filtering* dilakukan mendapatkan angka 97%.

Pada data nomor 82 dan 100 nilai *round trip time* dari *rogue access point* akan lebih kecil jika menggunakan angka yang didapatkan dari perhitungan *round*

trip time tanpa dilakukan *filtering*. Jika tidak menggunakan *filtering* yang terjadi adalah kesalahan pendeteksian *rogue access point*. Pada data nomor 82, nilai yang didapatkan oleh program pada perhitungan pada *legitimate access point* sebelum dilakukan *filtering* adalah 33,82 ms sementara nilai *round trip time* pada *rogue access point* didapatkan angka 32,52 ms. Setelah dilakukan *filtering* nilai yang dianggap anomali didapatkan nilai *round trip time* 12,5 ms untuk *legitimate access point* dan 12,875 ms untuk *round trip time* pada *rogue access point*.

Tanpa proses *filtering* data nomor 82 akan menimbulkan kesalahan deteksi karena nilai *round trip time* yang didapat dari *rogue access point* lebih kecil daripada nilai *round trip time* dari *legitimate access point*. Hal tersebut juga terjadi pada data pengujian yang didapatkan pada data nomor 100. Kenaikan akurasi yang didapatkan adalah 2%. Untuk menguji proses *filtering* benar-benar berjalan dengan baik digunakan *tool network shaper* dari *Netem* untuk melakukan simulasi jaringan yang tiba-tiba lambat.

Netem digunakan untuk menambahkan *delay* paket dengan nilai tertentu dan dalam waktu tertentu. Pada pengujian yang dilakukan, dengan menggunakan Netem untuk menambahkan *delay* sebesar 500 ms dengan jeda waktu tertentu. Hasil yang didapatkan nilai *round trip time* pada saat perhitungan pada *legitimate access point* menjadi melonjak tinggi. Namun, fungsi *filtering* yang dibuat mampu menghapus nilai *round trip time* yang dari *legitimate access point* yang tiba-tiba melonjak tinggi.

Pada **tabel 6.4** diperlihatkan hasil pengujian proses *filtering* yang dilakukan dengan menambahkan *delay* dengan menggunakan *Netem*. Pada 5 kali pengujian, ditambahkan 500 ms *delay* saat program melakukan perhitungan *round trip time* pada *legitimate access point*. Dalam 5 kali pengujian, pendeteksian *rogue access point* dengan *filtering* mampu memperbaiki nilai rata-rata *round trip time* awal. Proses *filtering* mampu menghapus nilai *round trip time* yang meninggi sebagai simulasi jaringan tiba-tiba melambat. Sehingga nilai *round trip time* yang dihasilkan oleh *legitimate access point* tetap lebih baik daripada *rogue access point*. Tanpa proses *filtering*, hal ini akan menimbulkan kesalahan pendeteksian.

BAB 7 KESIMPULAN DAN SARAN

Bagian ini akan memuat kesimpulan dan saran terhadap hasil implementasi dan pengujian yang dihasilkan dari sistem pendeteksian *rogue access point* dengan metode perhitungan *round trip time*.

7.1 Kesimpulan

Dari hasil implementasi dan pengujian sistem pendeteksian *rogue access point* dengan metode perhitungan *round trip time* yang telah dilakukan oleh peneliti dapat disimpulkan bahwa :

1. Untuk melakukan pendeteksian *rogue access point* dapat dilakukan dengan melakukan perbandingan waktu *round trip time* dengan cara mengirim paket DNS *lookup* pada kedua *access point* yang identik atau dicurigai sebagai *rogue access point*. *Rogue access point* akan menghasilkan nilai *round trip time* yang lebih tinggi sebagai dampak dari penambahan jumlah *hop* yang dilalui oleh *rogue access point.* Jumlah *hop* yang dilalui oleh *rogue access point* akan lebih banyak daripada *legitimate access point*. Implementasi pendeteksian *rogue access point* dilakukan dengan membuat program yang mampu melakukan pengiriman paket DNS *lookup*, melakukan perhitungan *round trip time* hinggan melakukan perbandingan dari nilai *round trip time* dari kedua *access point* yang dicurigai sebagai *rogue access point*.
2. Nilai *round trip time* yang dihasilkan oleh *rogue access point* akan selalu lebih tinggi secara teori akibat dari jumlah *hop* yang lebih banyak. Pada pengujian yang dilakukan pada jaringan pertama didapatkan rata-rata *round trip time rogue access point* mencapai 18,56471 ms berbanding 12,04939634 ms sebagai rata-rata *round trip time* yang didapat dari *legitimate access point*. Selisih yang dihasilkan mencapai 6,515313659 ms. Pada pengujian yang dilakukan pada jaringan kedua didapatkan rata-rata *round trip time rogue access point* mencapai 15,958 ms berbanding 9,543182927 ms sebagai rata-rata *round trip time* yang didapat dari *legitimate access point.* Selisih dari kedua *access point* yakni 6,414817073 ms. Kondisi jaringan yang tidak stabil akan mempengaruhi nilai *round trip time* yang dihasilkan.
3. Dari pengujian sistem pendeteksian *rogue access point* dengan metode perhitungan *round trip time* yang telah dilakukan, didapatkan hasil akurasi mencapai 90% dalam pengujian pada jaringan pertama. Dan dalam pengujian pada jaringan yang kedua dihasilkan akurasi sebesar 95%. Keakuratan proses pendeteksian terpengaruh oleh stabil atau tidaknya kondisi jaringan. Proses *filtering* yang dilakukan mampu meningkatkan akurasi dari 95% menjadi 97%. Proses *filtering* menghapus nilai *round trip time* yang anomali sebagai dampak kondisi jaringan yang tidak stabil.

7.2 Saran

Saran yang dapat penulis sampaikan untuk penelitian lebih lanjut mengenai sistem pendeteksian *rogue access point* dengan metode perhitungan *round trip time* adalah sebagai berikut:

1. Program pendeteksian hanya dapat diimplementasikan pada linux karena menggunakan perintah *dig* pada *command line* linux untuk melakukan DNS *lookup*. *Query time* yang dihasilkan perintah *dig* sesuai dengan waktu *round trip time* dari DNS *lookup* yang dihitung secara manual dari tangkapan *wireshark*. Diperlukan mekanisme lain untuk mengirim DNS *lookup* yang mampu berjalan pada windows, untuk menghasilkan nilai *round trip time* sebagus yang dihasilkan oleh perintah *dig* agar program mampu dijalankan di sistem operasi *windows*.
2. Diperlukannya penelitian lebih lanjut dalam mengatasi permasalahan kondisi jaringan yang lambat secara terus menerus sehingga menghasilkan nilai *round trip time* yang buruk dan mempengaruhi tingkat akurasi program.

DAFTAR PUSTAKA

Abdillah, M.F., 2011. Pengembangan Dan Uji Kinerja Sistem Pendeteksi Rogue Access Point Menggunakan Aplikasi Berbasis Web Dengan Metoda Pengukuran Waktu Round Trip Time. Jakarta: Universitas Indonesia.

Han, H., Sheng, B., Tan, C.C., Li, Q. dan Lu, S., 2009. A Measurement Based Rogue AP Detection Scheme. Nanjing: Nanjing University.

Han, H., Sheng, B., Tan, C.C., Li, Q. dan Lu, S., 2011. A Timing-Based Scheme for Rogue AP Detection. IEEE Transactions On Parallel And Distributed Systems, Vol. 22, No. 11.

Karygiannis, T dan Owens, L., 2002. Wireless Network Security 802.11, Bluetooth and Handheld Devices. Gaithersburg: National Institute Of Standards and Technology.

Lamping, U. dan Warnicke, E., 2014. Wireshark User's Guide: For Wireshark 2.1.

Nikbakhsh, S., Manaf, A.B.A., Zamani, M. dan Janbeglou, M., 2012. A Novel Approach for Rogue Access Point Detection on the Client-Side. Kuala Lumpur: Universiti Teknologi Malaysia.

Song, Y., Yang, C. dan Gu, G., 2010. Who Is Peeping at Your Passwords at Starbucks? To Catch an Evil Twin Access Point. Texas: A&M University.

Watkins, L., Beyah, R. dan Corbett, C., 2007. A Passive Approach to Rogue Access Point Detection. Atlanta: Georgia State University.